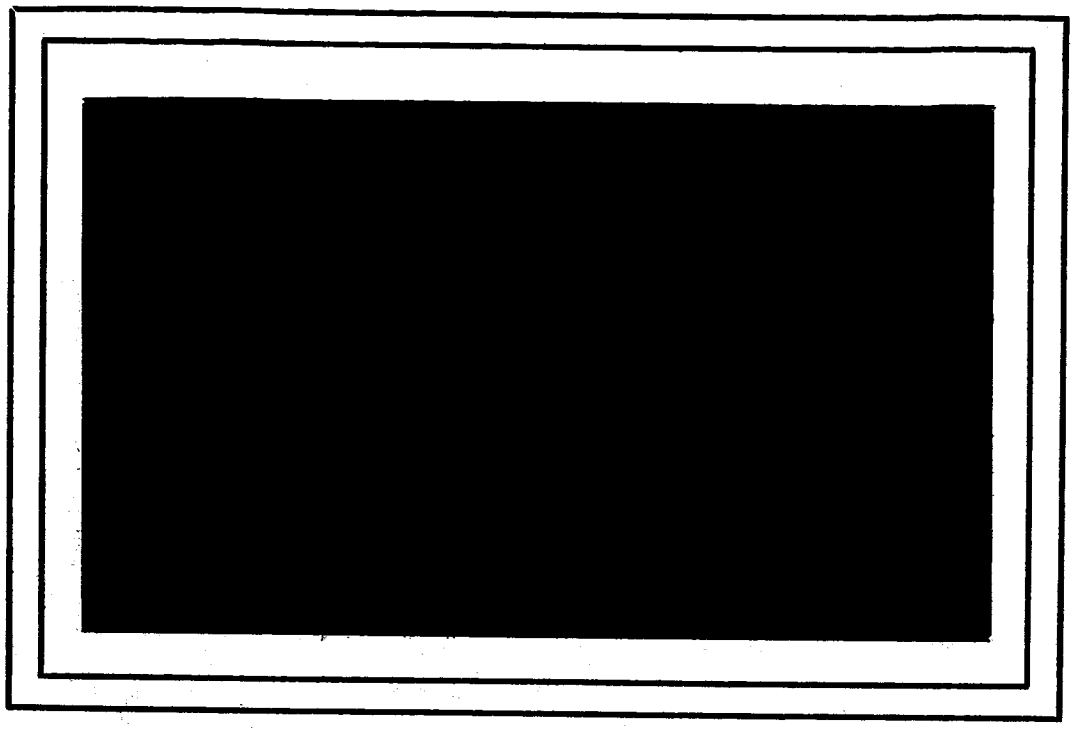

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

Technical Report TR-67-39
NsG-398 January 1967

BOOLEAN TRANSLATION OF THE SYMBOLIC DESIGN OF A DIGITAL COMPUTER

by

Bruce D. McCurdy
Department of Electrical Engineering
University of Maryland

This report constitutes a thesis submitted to the Faculty of the Graduate School of the University of Maryland in partial fulfillment of the requirements for the degree of Master of Science. Thesis supervisor was Professor Yaohan Chu, Computer Science Center and Department of Electrical Engineering.

ABSTRACT

This thesis is concerned with developing a set of five computer programs, for use with the IBM-7090/7094 computer, to simulate and translate a chosen computer design in Chu's Computer Design Language (CDL) into boolean (or logic) equations. The machine which is simulated and translated by this set of programs is a simple, non-trivial, stored-program digital computer. The programs are written in the Michigan Algorithmic Decoder (MAD) programming language.

The computer design (shown in Appendix A) is first transcribed into a MAD program, following the CDL format as closely as possible. This MAD program is readily expanded into a simulation program (shown in Appendix B). The simulation program uses as data a sample test program encompassing the entire instruction repertoire of the designed computer, and produces an octal output which represents all register contents for each clock (Appendix C).

The translation is accomplished by a set of four consecutive programs: a tabulation program, a truth table program, a boolean expression program, and a boolean equation program. The tabulation program (Appendix D) accepts as input the MAD program obtained above, prior to its expansion into the simulation program, and produces as output a tabulation of all data pertinent to translation (Appendix E).

This tabulation is then processed by the truth table program (Appendix F), which generates truth tables for each register and each operation (Appendix G). With these tables as input, the boolean expression program (Appendix H) produces as output a set of JK-type boolean expressions for each register flip-flop (Appendix I). The boolean equation program (Appendix J) assembles and packs these expressions into a set of boolean equations (Appendix K). These equations constitute the boolean translation of the chosen computer design.

The use of MAD to simulate a CDL design has both advantages and disadvantages. The advantages include: the use of subscripted alpha-numeric statement label vectors to simulate command signals; the ability to define arithmetical operators as external functions, and use them directly in the program without a separate calling sequence; the ability to simulate parallel conditional operations through the use of the MAD compound conditional; and the ability, by proper mode declaration, to use octal integers in all registers throughout the program. The disadvantages include: the inability to simulate adequately multiple clock pulses and matrix-generated command signals with the statement label vector; the necessity to right-adjust all integer values, which presents difficulties in bit-comparisons and sub-register operations; and the 36-bit word size required of all variables in MAD, by which it is difficult to simulate any smaller or larger register and memory word sizes used in the CDL design.

Additionally, it must be noted that this simulation process is very slow. For this particular computer simulation, compilation time was 20.4 seconds, and execution time was 1.8 seconds.

ACKNOWLEDGMENTS

The computer time for this project was supported by National Aeronautics and Space Administration Grant NsG-398 to the Computer Science Center of the University of Maryland, under Project Number 305/01/107.

The data for the Logical Simulation program consists of a sample test program for another similar computer developed by Professor Yaohan Chu in Chapter 11 of his book Digital Computer Design Fundamentals (McGraw-Hill Book Company, 1962).

TABLE OF CONTENTS

LIST OF TABLES

SECTION I: INTRODUCTION

1. Purpose. The purpose of this paper is an attempt to develop a translation program for Chu's Computer Design Language, hereinafter referred to as the CDL. This particular program uses as input the CDL description of a simple, hypothetical, general-purpose digital computer, which is listed in Appendix A. A second purpose is to investigate the feasibility of using CDL format to develop a logical simulation program.

2. Programming Language. The programming language used in this investigation is the Michigan Algorithmic Decoder language, hereinafter referred to as MAD. This language was originally chosen primarily because it was the only programming language of which this writer had any working knowledge. Upon further investigation, it was found to be a fortuitous choice, especially in the realm of logical simulation. One basic reason for this is that both the CDL and MAD are ALGOL-based languages, hence the similarity.

The MAD language allows abbreviations of certain standard expressions, a partial listing of which is shown in Table 1. These abbreviations are used frequently in the ensuing program development, both for simplicity, and as format restrictions for input to the program.

3. Procedure. The development of this program, which is fully discussed in the balance of the paper, is as follows:

a. Since the CDL terminology and symbols are not computer compatible, the CDL description is first transcribed into MAD

Table 1: MAD Abbreviations and Corresponding Expressions

Abbreviation	Expression
C'E	CONTINUE
D'N	DIMENSION
E'L	END OF CONDITIONAL
E'M	END OF PROGRAM
E'N	END OF FUNCTION
E'O	ENTRY TO
F'N	FUNCTION RETURN
L'T	LOOK AT FORMAT
N'S	NORMAL MODE IS
O'E	OTHERWISE
O'R	OR WHENEVER
P'N	PROGRAM COMMON
P'T	PRINT FORMAT
R'T	READ FORMAT
T'H	THROUGH
T'O	TRANSFER TO
V'S	VECTOR VALUES
W'R	WHENEVER

language, following the CDL format as closely as possible. This transcription is readily expanded into a simulation program, which is discussed in Section II.

b. The next step is the construction of truth tables, from which the boolean equations can be developed. This is discussed in Section III, and is, in fact, two successive programs: a tabulation program, and a truth table program. The input to the tabulation program is the MAD transcription of the CDL description, extracted from the body of the simulation program. The output of the tabulation program is in the form of punched cards which are introduced as data for the truth table program. The output of the latter program is also in the form of punched cards.

c. The final step is the generation of boolean equations for each register, and for a representative memory word, or row. All registers, and the memory, are assumed to be composed of JK flip-flops, and the resulting equations are therefore of the JK form. This portion is subdivided into two programs: a boolean expression program, and a boolean equation program. The data for the boolean expression program are the truth tables generated by the truth table program, in the form of punched cards. The output is also in the form of punched cards, and is used as input to the boolean equation program. The final output is a printout of a series of JK-type boolean equations for each of the flip-flops of the computer.

4. Conclusions. The conclusions to be drawn from the work discussed in this paper can be described as both retrospective and prospective.

a. The simulation program indicates that CDL is well suited to logical simulation, but requires transcription into an existing

programming language, in this case the language being MAD. Lest it be concluded that MAD could be used just as easily for simulation, it must be pointed that the marked similarity between the two languages in this particular case is more a coincidence than a reality, resulting mainly from the simplicity of the computer described. MAD may greatly resemble CDL, but they are not equivalent. The following points must be emphasized, for clarity:

i. A CDL command signal is the output of a logical switching matrix. A MAD statement label, on the other hand, is a vector, with single subscripts. A single clock pulse produces a one-row matrix, which is equivalent to a vector, and the statement label suffices in the simulation. A multiple clock pulse machine, however, produces a multiple-row command signal matrix, which can not adequately be described by a vector without loss of format similarity.

ii. A register, considered as a word, or variable, in MAD, can not be decomposed into sub-registers. To be sure, it may be masked, and the results may be shifted as necessary for proper positioning in a memory word. The resultant word, however, is not in reality a sub-register, but a separate distinct variable, in addition to the original register, and not a part of it. By the same token, cascading of registers, a simple procedure in CDL, is rather cumbersome in MAD. It requires several shifts and logical-or operations, with the result again being a new and distinct variable, larger than and separate from its supposed components. It should be fairly evident that a MAD simulation program for a more

sophisticated computer than the one described herein would scarcely resemble the CDL description, let alone be equivalent.

iii. Individual bits of a register, as used in CDL, are actually small sub-registers, and subject to the same restrictions. The problem would be magnified if several distinct and non-adjacent bits were used to generate a command signal, as might be the case in micro-programming.

iv. MAD uses a word length of 36 bits, while CDL has no such restriction. Consequently, the use of MAD to simulate registers of shorter word length would tend to waste memory, and cause difficulties similar to those encountered in sub-register operations, since all words must be right-adjusted. Words of greater length than 36 bits, on the other hand, would require decomposition and cascading, with the attendant difficulties already discussed.

In view of the above, it is felt that a CDL compiler, to accept a CDL description directly and produce a machine language simulation program acceptable to the simulating computer (in this case, the IBM-7090/7094), is both desirable and necessary.

b. The translation program developed in this paper can only be said to work for this particular CDL description. It does, however, demonstrate that such a translation is possible, and the basic ideas could be expanded for other more sophisticated computer descriptions. The following points should be considered:

i. If a CDL compiler should be developed, the tabulation program developed herein could easily be bypassed, if not

eliminated. Thus, the truth table program could readily develop the necessary truth tables directly from the description.

ii. The truth tables are a necessary intermediate step in the translation, since the boolean expressions developed from them would be entirely dependent upon the type of flip-flop used. However, the boolean expression and boolean equation programs could probably be combined, with the expressions being packed into the appropriate equation matrices as they are generated from the truth tables. Unfortunately, time did not allow investigation of this possibility.

SECTION II: LOGICAL SIMULATION

A. ADVANTAGES OF MAD LANGUAGE

There are several advantages to using MAD language in writing a Logical Simulation Program for a computer description written in CDL. These are:

1. Alphanumeric statement labels. A statement label effectively simulates a command signal in CDL, since it identifies the location of the next sequential instructions to be executed in the program. Since a MAD statement label is an alpha-numeric constant of not more than six characters, beginning with a literal, and, furthermore, may be subscripted, a CDL decoder terminal is easily transcribed into a MAD statement label. For example, K(17)*P in CDL becomes K(17K) in MAD (K denotes an octal subscript).

2. Integer mode. By declaring the normal mode to be integer (N'S INTEGER), all variables and constants not otherwise declared are manipulated as integers, either octal or decimal numbers. This [illegible] facilitates register manipulation.

3. Octal numbers. By appending a suffix, K, to any integer, it is automatically declared an octal integer. Thus all registers contents may be declared or manipulated as octal numbers, easily convertible to binary form.

4. Statement label vector. A vector declared in statement label mode can be subscripted with another integer variable, whose value may be evaluated at any specified point in the program, and the value may be an octal or decimal integer. Thus, evaluating the statement T'O K(F),

a transfer is made to statement label K(17K), for example, only when the value of F at that time is 17 octal (1111 binary).

5. Functions. In MAD, an external, or internal, function is defined by an alpha-numeric name of not more than six characters, followed by a period (.), and by the arguments in parentheses immediately following. It may be used as part of an arithmetic expression without an additional execution statement. Thus, the MAD statement, A = ADD.(A,R), evaluates the function ADD.(A,R), and substitutes the result as the value of variable A. In this way, the arithmetic and functional operators of the CDL are readily transcribed into MAD external functions, as shown in Table 2.

6. Substitution operation. The large majority of operations described in the CDL are transfer operations, defined by the symbol " ← ". This is identical to the substitution operation in MAD, defined by the symbol " = ". Thus the CDL statement, C ← D, readily becomes the MAD statement, C = D, without loss of meaning or format.

7. Conditional statements. By use of a compound conditional, several executable statements may be defined in a MAD program directly after the conditional statement which determines their execution. Thus parallel conditional operations can be simulated without loss of context or sequence. An example is shown in Table 3.

8. Masking. Using the bit-wise logical-and operation in MAD (.A.), together with an octal integer (readily converted to its binary value), any portion of a register can be masked in a MAD program, to evaluate, compare, or manipulate one or more bits.

Table 2: Arithmetical and Functional Operators

CDL	MAD Function
A add R	ADD.(A,R)
A sub R	SUB.(A,R)
1 shr A	SHRA.(A)
1 cirl A	CIRL.(A)
D count +1	UCOUNT.(D)

Table 3: Use of MAD Compound Conditional

CDL Operation	MAD Transcription
IF G = 0 THEN BEGIN C ← 0;D ← 0 END	W'R G.E.0 C = 0 D = 0
IF G = 1 THEN F ← 6	O'R G.E.1 F = 6K E'L

B. RESTRICTIONS OF MAD LANGUAGE

In using a MAD program for CDL simulation, there are several restrictions and difficulties which limit the possibility of exact transcription from CDL into MAD in the following operations:

1. Sub-register manipulation. In CDL, subregisters may readily be defined and manipulated as separate entities. This is difficult, if not impossible, in MAD. We will use the example in Table 4 to discuss this in detail.

   a. C←R(ADDR). In this operation, the address portion of the R register, R(ADDR), is transferred to the C register. This portion, or subregister, is defined as the last six bits, R(3) through R(8), of R. The operation is fairly straightforward in the MAD program, but it loses its similarity to CDL in the process. The MAD expression is C = R.A.77K, which means that the last six bits are extracted by masking with 77 octal (111111 binary). Since the result remains right-adjusted, no further manipulation is required.

   b. F(I)←R(OP). In this operation, the operator portion of the R register, R(OP), is transferred to the instruction portion of the F register, F(I). These subregisters are defined as the first three bits of R, R(0) through R(2), and the last three bits of F, F(1) through F(3), respectively. The corresponding MAD operation is not so simple. Simply masking the first three bits of R with 700 octal (111000000 binary) extracts the required bits, but with six trailing 0's. The result must now be right adjusted by a six-bit shift. The MAD expression thus becomes (R.A. 700K).RS 6.

Table 4: Sub-register Transfers in CDL and MAD

CDL Operation	MAD Transcription
C ←R(ADDR)	C = R.A.77K
F(I) ←R(OP)	F = ((R.A.700K).RS6).A.7K
F(0) ←0	

Table 5: Bit Comparison in CDL and MAD

CDL Operation	MAD Transcription
IF C(3) = 1 THEN BEGIN F ←17;G ←0 END	W'R (C.A.4K) NE.0 F = 17K G = 0
IF C(2) = 1 THEN F ←16	O'R (C.A.10K).NE.0 F = 16K
IF C(1) = 1 THEN F ←15	O'R (C.A.20K).NE.0 F = 15K
IF C(0) = 1 THEN F ←14	O'R (C.A.40K).NE.0 F = 14K
	E'L

c. F(0) ← 0. In this operation, the first bit of F, F(0), is set to 0. In the MAD program, since F is to be considered a four-bit register, performing such an operation separately would cause further complications. It is therefore considered more desirable to combine these two CDL operations into one MAD operation, where F(0) is set to 0 by masking F with 7 octal (111 binary). The resultant expression is: F = ((R.A.700K).RS.6).A.7K.

2. Bit comparison. In CDL, the logical value of a specific bit may be used as a condition for execution of a transfer statement. The bit may be masked for evaluation in MAD, but unless it is the right-most bit, the result is an octal number with trailing 0's, and not a logical value (0 or 1). The result may be right-adjusted by shifting; it may be compared with an octal number; or, if the logical value is to be 1, it may be tested for inequality with 0. This simulation program uses the latter operation, as shown in Table 5.

3. Clock pulse simulation. In CDL, a command signal is not generated until a clock pulse activates the appropriate decoder terminal; e.g. K(17)*P. Such an expression is illegal in a MAD statement label, which is used to simulate the command signal, as already discussed. In the case of a single clock pulse, as used in this program, the problem is minor. An external function, CLOCK.(P), executes a transfer to statement label K(F), with the subscript being the octal value of F at the time of execution. In more sophisticated machines, with variable or multiple clocks, the CLOCK. function must, of necessity, be more complicated, and the advantage of the MAD statement label may be lost.

C. SIMULATION PROGRAM PROCEDURES

The simulation program of a simple digital computer, listed in Appendix B, is essentially a MAD language transcription of the CDL description, incorporating the advantages, and subject to the limitations listed above. The following additional features will now be discussed.

1. Declarations. The declaration portion of the CDL description is not essential to the MAD simulation program, but is necessary, directly or indirectly, for a translation program. It is therefore included as MAD comments, identified by the letter R in column 11. Any additional comments, including those in the CDL description, can be inserted in the same way. The MAD declaration portion has no counterpart in the CDL description, but is essential for a MAD program.

2. Additional functions. The clock simulation function, CLOCK.(K,F), has already been discussed, and is considered self-explanatory. An additional function, PRINT.(P), has been defined, to print out the octal contents of each register after each sequential operation, so as to visually check the sequencing and proper operation of the computer according to the program introduced.

3. Cycling simulation. In the actual operation of this computer, when the contents of F are 17 octal (1111 binary), and G is 0, the C and D registers are reset, or cleared, and the computer "cycles", or does not move until G is set to 1. This operation occurs twice: when the power is turned on, and when the stop instruction, 504 octal (10100100 binary), is executed. To simulate this in the MAD program, the following procedure is used:

a. An additional argument, WAIT, is introduced for the external

function, PRINT, at statement label K(17K): PRINT.(P,WAIT).

b. When the "cycling" conditions occur in the external function (F = 17K and G = 0 and C = 0 and D = 0), the statement ERROR RETURN is executed. Since the external function has only one dummy argument, this statement causes a return to the main program, not at the original point of calling, but at the statement label identified by the second argument in the function call, namely WAIT.

c. At the statement label WAIT, two conditions may occur:

i. When the sequential number of the clock pulse is less than 5, (P.L.5), it is assumed the power has just been turned on, and the START switch is to be turned on to start the program operation. This is simulated by the statement T'O START, and START is printed in the simulation output.

ii. Otherwise (O'E), it is assumed that the "stop" instruction has been executed, STOP is printed in the output, and the program terminates.

4. Program. The program for this simulation is introduced as data. Instructions and operands are obtained by transferring a memory word, located by the memory address register, C, into the buffer register, R, with the "fetch sequence" operation: R M(C). The memory is declared in the MAD program as an integer vector with 64 components, or nine-bit words. The data therefore consists of a sequence of memory components, e.g. M(63), with the value of each being an integer of three octal digits (nine binary bits). Unfortunately, vectors introduced as data can not have octal subscripts, and the memory address locations, determined by C, must be converted to decimal form for data purposes.

5. Simulation output. The output for this simulation is an octal representation of the register contents, according to the format specified in the MAD simulation program. An additional statement in the PRINT.(P) function (P'T STATE,F), together with a carriage control character of " + " in the format, which causes an overprint of the same line, allows the command signal for each operation to be printed according to the octal value of F after the preceding operation; e.g. K(17). The data, also printed out (READ AND PRINT DATA), was developed by Chu for another digital computer, essentially the same as the one described here. The program uses the full instruction repertoire of this computer, and the results have been verified to be accurate. A listing of the output of this program is found in Appendix C.

SECTION III: TRUTH TABLE GENERATION

A. TABULATION

1. Purpose. The purpose of this program, shown in Appendix D, is to scan the simulation program, which is now a MAD transcription of the CDL, to extract the information pertinent to translation into Boolean equations, and to tabulate this information into a usable format.

2. Data. The data for this program is the execution portion of the simulation program discussed in the previous chapter, less the CLOCK.(K,F), and PRINT.(P) statements. Inasmuch as the CDL must be transcribed into MAD language for simulation, it is felt that no undue difficulty would be imposed by establishing a specific format for this data. Therefore, the following restrictions are imposed:

a. All statement labels must start in column 1.

b. All statements, conditional or unconditional, must start in column 12.

c. Except for the conditional jump instruction, all conditionals will be compound conditionals.

d. All blanks, or spaces, will be eliminated in substitution statements, boolean expressions, and arithmetic expressions.

e. In all conditional statements, the MAD abbreviation will be used exclusively.

f. In compound conditionals, the executable statements following the conditional will begin in column 16.

3. Input Format. The input data, subject to the conditions cited above, is read in using two formats: IN1, for conditional statements,

and IN3 for unconditional statements.

4. Procedure. Using the LOOK AT FORMAT (L'T) statement, a card is checked for the contents of columns 12 through 14. Several alternatives occur:

a. If the contents are W'R or O'R, a conditional statement exists. The name and value of the conditional variable are determined (SWV and VAL), as is the statement label, or command signal name (STATE). If SWV is A(0), and there is a simple conditional, the register involved (REG), and its final contents (CONT), are evaluated. If there is no simple conditional, the next card is read. Since, by the format, columns 12 through 14 are now empty, REG and CONT are evaluated from the contents of columns 16 through 33 (three 6-character words). The five variables so obtained are now printed out according to the format OUT. STATE, SWV, and VAL are now "blanked", and the next card checked by L'T IN1, for an entry in STATE or SWV. If not, parallel operations are occurring. New REG and CONT are evaluated, and printed out (with blanks in STATE, SWV, AND VAL). If the STATE or VAL columns are not empty, the procedure begins again from the original L'T statement, at statement label START.

b. If columns 12 through 14 are E'L, no action is taken, and the program returns to START.

c. If columns 12 through 14 are O'E, an error has occurred in the input, and the program terminates.

d. E'M or END in columns 12 through 14 indicate the last card of the input, and the word END is printed out and the program terminates.

e. Any other characters in columns 12 through 14 causes the IN3 format to be used to read in data at statement label NOCON. This reads STATE from column 1 through 5, and columns 12 through 29 are read as three 6-character words, from which REG and CONT are evaluated. Since this is an unconditional statement, SWV and VAL are not considered in printing out the format OUT 2.

5. Special conversions. In order to match the output to the CDL format as closely as possible, several special conversions are made in this program:

a. Whenever CONT is R.A.77, this is converted to R(ADDR, both terms being truncated to six characters for easier manipulation.

b. Whenever REG is F and CONT is ((R.A.7, this is printed out by a special pair of formats, FI and FO, into its original terms, with the transfer operator omitted but always implied: F(I) ← R (OP), and F(0) ← 0.

c. Whenever column 12 contains M, this indicates the store instruction, and REG becomes M(C).

6. Output. The output from this program, printed according to the formats described above, is shown in Appendix E. When punched onto cards, it becomes the input data for the next program.

B. TRUTH TABLES

1. Executive Program. This program reads in the data consecutively, and stores it into five vectors, corresponding to the five input variables generated by the previous program: STATE, SWV, VAL, REG, and CONT. Where parallel operations occur, the corresponding values of STATE, SWV, and VAL are carried through for each operation. The

"loading" loop is terminated when the END card is reached.

2. F (Control) Register program (FTAB). This program is an external function of the above executive program, and is the first one executed after the "loading" loop is terminated. It searches the input vector, and performs operations only when the "character" value of REG is F or F(I). For each operation, the subscript of K stored in STATE is evaluated as a two digit octal number, and is converted to four binary bits, X(0) through X(3), corresponding to each flip-flop of the F register. The succeeding operations, determined by whether REG is F or F(I), are:

a. When REG equals F, the column headings are printed according to format HEAD1. The starting values of F(0) through F(3) have already been determined, while the value of SWV, YY, is determined from VAL. The final values of the F-bits, Z(0) through Z(3), are computed by converting the two digit octal value of CONT to binary form. These are printed out by format OUT1.

b. When REG equals F(I), the column headings are printed according to format HEAD, including the three bits R(0) through R(2). The starting values of the F-bits have already been determined, the final value of F(0), ZZ, is set to 0, and the final values of F(1) through F(3) are determined by the starting values R(0) through R(2), computed as Y(1) through Y(3). These are computed by naming a new variable, ROP, and converting its value into three binary bits. The entire process is repeated six times, as the value of ROP varies from 0 to 5, corresponding to the instruction repertoire of this computer. These values are printed out by the format OUT2.

c. In both of the above cases, the output is in the form of a one-line truth table for each set of column headings.

3. Accumulator program (ATAB). This program, or external function, generates three separate types of truth tables, according to the value of CONT. In addition, in the shifting operations, the first and the last bit operations may differ from the other bits. Three sets of truth tables must therefore be generated; for the first bit, A(0), the last bit, A(8) (A is assumed as nine-bits by the CDL declaration), and a representative bit, A(I), where I will vary from 1 to 7.

a. If CONT equals 0, this is a clear operation, and requires a one-variable truth table. The column headings, defined in the program, are printed according to format CLR, and the values are generated and printed by the external function PRNT1.(BB), where the final value of the variable, BB, is declared as 0.

b. If CONT equals "SHRA.(" or "CIRL.(" (truncated to six characters), this is a two variable, or transfer, truth table, with the final value of the first variable being the starting value of the second variable. The column headings, determined by the direction of the shift, and defined in the program, are printed according to the format TRF. The values are computed and printed by the external function PRNT2.(BB).

c. If CONT equals "ADD.(A" or "SUB.(A", this calls for a truth table of three input variables (A,R, and the carry bit, B) and two output variables, (A and B). This section of the program sets two boolean variables, AY and AX, as switches, to determine whether the operation is addition or subtraction, (AX), and to

determine if the A(0) table is being generated, (AY), in which case the carry output column is suppressed. The column headings are printed according to format ADSUB, and the values are generated and printed by the external function PAS.(AX,AY). This function performs full binary addition or subtraction, according to AX, generating a final value of A and a carry bit.

4. Operation counter (D) register program (DTAB). This external function generates three types of truth tables:

a. When CONT equals 0, the column headings, as defined in the program, are printed out by format CLR, and the values by external function PRNT1.(BB), with BB declared as 0.

b. When CONT equals "R(ADDR", the column headings are printed by format TRF, and the values by external function PRNT2.(BB).

c. When CONT equals UCOUNT, this is a six-bit counter truth table, with the column headings printed by format HEADD1, and the values computed and printed by external function PCNT.(AZ).

5. Memory address (C) register program (CTAB). In this external program, two types of truth tables are generated:

a. When CONT equals 0, column headings, defined in the program, are printed according to format CLR. PRNT1.(BB) generates a one variable truth table with BB declared as 0.

b. When CONT equals "D" or "R(ADDR", this is a transfer operation. Column headings are printed according to format TRF, and a two variable truth table is generated by PRNT2.(BB).

6. Start/Stop flip-flop (G) program (GTAB). This register, or flip-flop, requires only one-variable truth tables generated by PRNT1.(BB). BB is declared as the final value of G, after the operation.

7. Buffer (R) register and memory programs (RTAB and MEMTAB). These truth tables are of the transfer, or two variable, variety. The memory truth table describes a representative bit, where the column is designated by I, a representative index varying from 0 to 8, and the row is designated by C, where C is undetermined, but represents the value of the memory address register, C.

8. Additional comments. In each of the above functions, except for FTAB, the input vectors are searched sequentially, by a loop, and the external function is executed only when the value of REG is equal to the applicable register. At this point, STATE and SWV (previously complemented if VAL = 0), are evaluated and printed out by format HEAD2. Thus, before each truth table (or tables, in the case of the accumulator), the control signal and conditional variable are printed.

9. Output. The output of this program is a series of truth tables, as shown in Appendix G, which are used as data for the next program portion.

SECTION IV: BOOLEAN EQUATION GENERATION

A. BOOLEAN EXPRESSIONS

This program generates the boolean expressions for each register bit, using JK flip-flops. The executive, or main, program will be discussed in three separate parts, followed by the external functions utilized.

1. Executive program (I). The input to the executive program is the output of the Truth Table Generation program, previously discussed. For this portion, the input format IN1 reads the command signal, or K-terminal signal (STATE), the four bits of the register, (F(0) through F(3)), and three possible switching, or conditional variables, (SWV(0) through SWV(2)). The next format reads the initial value of the F-bits (F(0) through F(3)), the value of the switching variable, or variables, (V(0) through V(2)), and the final value of the F-bits (FN(0) through FN(3)). Then the boolean expressions are generated by the external function VBLF.(NX), where NX is a dummy variable of no consequence, for calling purposes only. This procedure is repeated until REG(0) is no longer F(0).

2. Executive program (II). The first card for this program is read according to format STS, at statement label NEXT, to evaluate the command signal, (STATE), and the conditional variable (SWV(0)) for the following operation. The next card is tested according to format DREG, to look for the counter operation. If this should occur, by the fact that DX(2) is equal to "D(2)", the card is read by the format DREG. The counter boolean expressions are generated by external function PCNTR.(NX). When control returns to the executive program,

a transfer is made back to statement label NEXT.

3. Executive program (III). If, by the DREG format test, DX(2) is not equal to "D(2)", the card is read by format RGSTR, at statement label A3. This format reads five possible column headings for the truth table. B(2), the third input column, is tested, and if it is not blank, an addition or subtraction truth table follows. The boolean expressions are generated by the external function VBL32.(NX). If B(2) is blank, B(0) is then tested. A blank in B(0) will call VBL1.(NX), while no blank will call VBL2.(NX), which read one- and two-variable truth tables respectively. Upon return to the executive program, the column headings are tested to determine if A(0) or A(I) truth tables have been generated. If so, the program transfers back to A3, where the next card is read according to format RGSTR. If the column headings indicate A(8) truth tables have been generated, this means that a new command signal (STATE) will be involved, and transfer is made to NEXT to read the next card by format STS. When all the input cards have been read, the program terminates.

4. F-register boolean expression generator (VBLF). This external function has access, through PROGRAM COMMON, to the values read in to STATE, REG(0)...REG(3), SWV(0)...SWV(2), F(0)...F(3), V(0)...V(3), and FN(0)...FN(3). The variable names in SWV(0) through SWV(2) are complemented with a "prime" (') if the correspohding V(0) through V(2) is 0. The initial and final values of the register bits are compared consecutively, for any change of state. If STATE is K(12), a special subroutine is involved, since F(0) always changes from 1 to 0, and the other bits change according to the corresponding value of R(0) through R(2). Otherwise, the register bit is determined (REG), the appropriate

flip-flop terminal involved (BIT), the command signal (STATE), and the switching variable (SWV). The boolean expressions are printed out by formats OUT and OUTO, and control is returned to the executive program.

5. One-variable truth table (VBL1.). This program reads the one-variable truth table as follows:

a. The command signal (STATE), the conditional variable (SWV), and the variable table heading (B(1)), are available by PROGRAM COMMON.

b. The starting and final values are read in for each row (two rows in this case) by format VAL1, and the two values compared. If an expression is to be generated, J or K is determined, and the "prime" symbol ('), if required, is generated.

c. If B(1) is "C(I)" or "D(I)", six sets of expressions are generated, with the representative index, I, becoming the indices 0 through 5. If B(1) is "A(I)", the expressions for A(1) through A(7) are generated. Otherwise, a single set of expressions is generated for the column heading involved.

d. If SWV is empty, the expressions are compressed and printed out by format OUT1, otherwise they are printed out by format OUT.

6. Two-variable truth table (VBL2.). The procedures here are similar to those for VBL1., with STATE, SWV, B(0), and B(1) as PROGRAM COMMON, except for the following:

a. The values corresponding B(0), B(1) and B(3) are read in by format VAL2, for each of the four rows, as X, Y, and Z.

b. The index in column B(1), now the second, or independent variable, may be indexed with a representative arithmetic expression such as: A(I-1), A(I+1), or R(I+3). In this case, as

I is incremented in the same way as in VBL1., the arithmetic indices are also evaluated accordingly.

c. If B(0) is "M(C,I)", this requires a different format, to allow for a six character name in B(0). Additionally, as before, compressed formats are required if SWV is blank. Consequently, there are four output formats: OUT4, OUTU4(compressed), OUT6(six characters in B(0)), and OUTU6(compressed.).

7. Addition/subtraction truth tables (VBL32.). This program reads truth tables with three inputs and two outputs. STATE, and B(0) through B(4) are PROGRAM COMMON (there is no SWV for these tables). Procedures are as follows:

a. The values of B(0) through B(4) are read in by format VAL3, as V(0) through V(4).

b. The J or K signal is determined by comparing the values of V(0) and V(3).

c. The complement symbol, or "prime" ('), is introduced for B(0), B(1), and B(2), when the corresponding V is 0.

d. If B(0) is equal to "A(0)" or "A(8)" the boolean expression is printed out by format OUTA.

e. If B(0) is equal to "A(I)", seven boolean expressions, with numerical indices, are printed out, for A(1) through A(7), using format OUTA.

f. Whenever B(4) is "B(I-1)", and the V(4) is 1, B(0), B(1), and B(2), with "primes" where necessary, and with numerical subscripts, are used to print out the eight boolean expressions for "B(0)" through "B(7)" (the carry-bits), using format OUTCY. The boolean expression for "B(8)" is then printed out by format CY8.

8. Counter boolean expressions (PCNTR.). Since, in this program, pre-determined expressions will be generated, the 64 truth value cards for the counter truth table have been extracted from the input, leaving only the column headings. These character values are carried through as DX(0) through DX(11) by PROGRAM COMMON, and the twelve boolean expressions (D(0)J through D(5)J, and D(0)K through D(5)K) are generated by the PCNTR.(NX) external function. Control is then returned to the main, or executive, program.

9. Output. The output of the boolean expression program is shown in Appendix I.

B. BOOLEAN EQUATIONS

The executive, or main, program for this portion will be discussed in two sections, in conjunction with the applicable external functions.

1. Executive program (I). Reading in the individual boolean expressions successively, using format IN, this portion terminates when the F register expressions have all been read. In the process, it loads the expressions, according to the loop index, into four vectors: BIT, STATE, CLOCK, and VAR. It has previously been determined, by inspection, that only one variable, other than the command signal, is involved in the F register equations. These vectors are declared as PROGRAM COMMON for ready access by the equation generating external function, FEQN.(A).

2. F-register equation generator (FEQN.(A)). The argument in this, and all successive equation generating functions, is the index of the "loading loop". In this function, eight vectors are declared: F(0)J through F(3)J, and F(0)K through F(3)K. These vectors are

declared as rows of an array, or matrix, called F. Each row has an index, N(1) through N(8), which keeps track of the next empty column position in each matrix row (or vector). The names are loaded into the first column, and the constant called EQUAL into the second column. The expressions are first minimized, and duplications are eliminated. Each expression, in turn, is checked for the corresponding F matrix row, and the three parts of each expression are loaded into the vacant rows of the appropriate vector, preceded by the constant called PLUS for all but the first expression. Each vector thus becomes a complete boolean equation. In addition, for the command signal POWER, the J terminal of each flip-flop is grounded, setting the flip-flop to 1. Thus, POWER is added as the last term of each J equation. The equations are printed out, using the format EQNS.

3. Executive program (II). This portion reads in the succeeding boolean expressions, already arranged by registers. Each expression is loaded and packed by the external function LOAD.(-,T), the missing argument being the index used in each successive "loading loop". The argument T is necessary in the case of the memory equations, where the flip-flop terminal name is a seven-character name, M(C,I)J or M(C,I)K, which exceeds the storage capacity of one IBM 7094 memory word. Thus, in this unique set of expressions, T is used to store the J or K indicator. Using the L'T IN procedure, the first letter of the expression name is checked, and each "loading loop" is terminated when this letter changes. After each termination, the appropriate external function is executed to generate the boolean equations for that register. After all equations have been generated, the program terminates.

4. Loading function (LOAD.(X,T)). Using the arguments as described above, this program first determines which read format to use, and reads in the boolean expression as the expression name, JK, and then forty two single characters, VX(0) through VX(41). These characters are compressed to eliminate blank characters, and loaded, with trailing blanks, into a seven-word vector, VY, by the external function CMPCK.(VX,VY). This vector is then loaded sequentially into a single column of a seven-row matrix, Y, the particular column being determined by the index generated in the executive program. Control is then returned to the executive program.

5. G-register equation generator (GEQN.(B)). This function defines a two-row matrix, each row being one of the G equations, G(0)J or G(0)K. The name is loaded into the first column, and EQUAL into the second column. Two indices, NG(1) and NG(2) keep track of the next empty column in each row. The vector, BIT, and matrix, Y, are scanned sequentially by columns, and BIT is matched with the first column of G, to determine the corresponding row. The same packing procedure is again used as in FEQN. However, POWER and START generate independent signals to the K and J terminals respectively, and are loaded as complete expressions, ignoring whatever else may have been included by the input expression. The two rows are then printed out according to format OUT. The results are the boolean equations.

6. Memory-word equation generator (MEQN.(X,T)). This function uses the second argument, T, to store the J or K of the boolean expression name, as has been explained in regard to LOAD(-,T) above. Otherwise, the equation packing procedures are the same as in FEQN. The matrix is M, the row index vector is NM, and the output format is OUTM.

7. D-register equation generator (DEQN.(X,T)). In the other registers, five memory words serve to store the entire boolean expression. Hence, only the first five components of the Y matrix column are loaded into the equation matrix row, while the other two components are ignored. In the D register, however, the D(0) counter expression requires the full seven words. Consequently, all seven Y components are loaded into the equation matrix row. Here the equation matrix is D, the indicator vector is ND, and the output format is OUTD. The first column of D, which has twelve rows, is leaded by inserting the index, 0 through 5, into the parentheses of the words "D(0)J" and "D(0)K" respectively, using a shifting of the index and a logical-or operation.

8. C-register equation generator (CEQN.(B)). The loading procedure in this function, including the equation name generation, is identical to the procedure for DEQN., except that only five components of Y are used. The matrix is C, twelve rows in size, the index vector is NC, and the output format is again OUT.

9. R-register equation generator (REQN.(X)). The matrix for this function is R; the indices are NR(1) and NR(2); the row, or equation names are R(I)J and R(I)K, and the output format is OUT. The procedures are in all respects similar to GEQN, except that the entire contents of BIT are matched to determine the appropriate row for loading.

10. A-register equation generator (AEQN.(X)). This function, in addition to generating the JK flip-flop equations for the A register, also generates carry-bit (B) equations for the addition and subtraction operations. The A matrix is defined with 18 rows: A(0)J through A(8)J,

and A(0)K through A(8)K. For format purposes, the indices are printed as decimal vice octal integers. For the carry bits, a B matrix is defined of nine rows, B(0) through B(8). The row, or equation, names are loaded using the same process as for DEQN. The BIT vector and Y matrix are read in sequentially by columns. BIT is compared to the twenty-seven row names to determine the appropriate equation row, and the expression is loaded into the row, according to the corresponding indicator, NA or NB. This is repeated until BIT and Y have been completely unloaded. Control is returned to the main program, which then terminates.

11. Error return. In each of the equation generating programs, an error return is executed whenever there is no match between BIT and the matrix row names for that program. Ideally this should never occur, but it serves to terminate the program early if one of the input cards should be out of sequence, and this was not detected in the main program.

12. Output. The output for this program is shown in Appendix K, and is the boolean translation of the original CDL description.

APPENDIX A

DESCRIPTION OF A SIMPLE DIGITAL COMPUTER USING CHU'S COMPUTER DESIGN LANGUAGE

```
COMMENT BEGIN THIS IS THE DESCRIPTION OF THE FIRST
VERSION OF A DESIGN OF A SIMPLE STORED PROGRAM DIGITAL
COMPUTER FOR THE PURPOSE OF ILLUSTRATING CHU'S COMPUTER
DESIGN LANGUAGE.  THE NINE-BIT INSTRUCTION FORMAT CONSISTS
OF A THREE-BIT OP CODE AND A SIX-BIT ADDRESS FIELD.
THERE ARE NINE INSTRUCTIONS:  ADD, SUBTRACT, CONDITIONAL
TRANSFER, UNCONDITIONAL TRANSFER, STORE ACCUMULATOR,
SHIFT LEFT, SHIFT RIGHT, CLEAR ACCUMULATOR, AND STOP.
THE SEQUENCING IS CONTROLLED BY OPERATION COUNTER F.
START-STOP IS CONTROLLED BY FLIPFLOP G.  END
REGISTER R(0-10); F(0-3); A(0-10); C(0-5); G(0); D(0-5).
SUBREGISTER R(OP)≡R(0-2); R(ADDR)≡R(3-10); F(I)≡F(1-3).
MEMORY M(C)≡M(0-77, 0-10).
DECODER K(0-17)≡F.
SWITCH PØWER(ØN,ØFF); START(ØN,ØFF)
CLOCK P
COMMENT BEGIN THE FOLLOWING STATEMENT WITH LABEL POWER(ON)
INDICATES WHAT HAPPENS WHEN POWER SWITCH IS TURNED ON
AND THAT WITH LABEL START(ON) INDICATES WHAT HAPPENS
WHEN THE OPERATOR NEXT TURNS ON THE START SWITCH.  END
```

```
POWER(ON):    F ←17; G ←0.
START(ON):    G ←1.
K(17)*P:      IF G=0 THEN BEGIN C←0; D←0  END;
              IF G=1 THEN BEGIN F←6.
K(06)*P:      R←M(C); D count +1;
              IF G=0 THEN F←17;
              IF G=1 THEN F←12.
K(12)*P:      F(I)←R(OP); C←R(ADDR); F(0)←0.
K(00)*P:      R←M(C); F←10.
K(10)*P:      A←A add R; F←13.
K(01)*P:      R←M(C); F←11.
K(11)*P:      A←A sub R; F←13.
K(04)*P:      D←R(ADDR); F←13.
K(02)*P:      IF A(0)=0 THEN F←13  END;
              IF A(0)=1 THEN BEGIN D←R(ADDR); F←13  END.
K(03)*P:      M(C)←A; F←13.
K(05)*P:      IF C(3)=1 THEN BEGIN F←17; G←0  END;
              IF C(2)=1 THEN F←16;
              IF C(1)=1 THEN F←15;
              IF C(0)=1 THEN F←14.
K(13)*P:      C←D; F←6.
K(14)*P:      A←1 shr A; F←13.
K(15)*P:      A←1 cirl A; F←13.
K(16)*P:      A←0;        F←13.
              END
```

APPENDIX B

SIMULATION PROGRAM

```
R COMPUTER SIMULATION, SEQUENCING CHECK


R                    DECLARATIONS

R  REGISTERS -       A(0-8),C(0-5),D(0-5),R(0-8),F(0-3),G(0)

R  SUBREGISTERS -  R(OP)    = R(0-2)
R                  R(ADDR)  = R(3-8)
R                  F(I)     = F(1-3)

R  MEMORY -          M(C) = M(0-63,0-8)

R  DECODER -         K(0-17K)  (NOTE - IN MAD LANGUAGE, K AFTER
R                               A DIGIT INDICATES AN OCTAL
R                               NUMBER, WHILE NO LETTER SUFFIX
R                               INDICATES A DECIMAL NUMBER)

R  CLOCK -           P

R  SWITCHES -        POWER(ON/OFF), START(ON/OFF)


R                    PROGRAM BEGINS HERE

 NORMAL MODE IS INTEGER
 PROGRAM COMMON G,C,D,R,F,A
 STATEMENT LABEL K,WAIT
 D'N K(15),M(63)
 P=0
BEGIN
 READ AND PRINT DATA
 PRINT  COMMENT  $1              COMPUTER SIMULATION, SEQUENCING
1CHECK$
 PRINT  COMMENT  $0                (ALL REGISTER CONTENTS ARE OCT
1AL)$
 P'T HEAD
 PRINT COMMENT $0                     POWER ON$
 PRINT COMMENT $ $

R***** BEGINNING OF OPERATION SEQUENCE

POWER
 F=17K
 G=0
 PRINT.(P)
 CLOCK.(F,K)

START
 G=1
 PRINT.(P)
 CLOCK.(F,K)
```

```
K(17K)      W'R G.E.0
                C=0
                D=0
            O'R G.E.1
                F=6K
            E'L
            PRINT.(P,WAIT)
            CLOCK.(F,K)

K(06K)      R=M(C)
            D=UCOUNT.(D)
            W'R G.E.0
                F=17K
            O'R G.E.1
                F=12K
            E'L
            PRINT.(P)
            CLOCK.(F,K)

K(12K)      C=R.A.77K
            F=((R.A.700K).RS.6).A.7K
            PRINT.(P)
            CLOCK.(F,K)

K(00K)      R=M(C)
            F=10K
            PRINT.(P)
            CLOCK.(F,K)

K(10K)      A=ADD.(A,R)
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(01K)      R=M(C)
            F=11K
            PRINT.(P)
            CLOCK.(F,K)

K(11K)      A=SUB.(A,R)
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(04K)      D=R.A.77K
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(02K)      F=13K
            W'R (A.A.400K).NE.0, D=R.A.77K
            PRINT.(P)
            CLOCK.(F,K)
```

```
K(03K)      M(C)=A
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(05K)      W'R (C.A.4K).NE.0
                G=0
                F=17K
            O'R (C.A.10K).NE.0
                F=16K
            O'R (C.A.20K).NE.0
                F=15K
            O'R (C.A.40K).NE.0
                F=14K
            E'L
            PRINT.(P)
            CLOCK.(F,K)

K(13K)      C=D
            F=6K
            PRINT.(P)
            CLOCK.(F,K)

K(14K)      A=SHRA.(A)
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(15K)      A=CIRL.(A)
            F=13K
            PRINT.(P)
            CLOCK.(F,K)

K(16K)      A=0
            F=13K
            PRINT.(P)
            CLOCK.(F,K)


           R***** SIMULATE CYCLING WITH G EQUAL 0

WAIT        W'R P.L.5
                PRINT COMMENT $0                    START$
                PRINT COMMENT $ $
                T'O START
            O'E
                PRINT COMMENT $0                    STOP$
            E'L

            V'S HEAD=$1H0,S5,5HCLOCK/1H ,S5,5HPULSE,S8,5HSTATE,S9,1HG,S4,
           11HC,S4,1HD,S4,1HR,S5,1HF,S4,1HA*$
            END OF PROGRAM
```

```
 EXTERNAL FUNCTION (X)
 NORMAL MODE IS INTEGER
 ENTRY TO SHRA.
R***** ARITHMETIC RIGHT SHIFT
 Y = X.A.400K
 Z = X.RS.1
 X = Y + Z
 FUNCTION RETURN
 ENTRY TO CIRL.
R***** CIRCULATING LEFT SHIFT
 Y = X.A.400K
 Z = (X.LS.1).A.777K
 X = Z + (Y.RS.8)
 FUNCTION RETURN
 ENTRY TO UCOUNT.
R***** INCREMENT COUNTER
 X=X+1
 FUNCTION RETURN
 END OF FUNCTION
```

```
 EXTERNAL FUNCTION (X,Y)
 NORMAL MODE IS INTEGER
 ENTRY TO SUB.
R *****(SUBTRACTION BY ADDITION OF 2'S COMPLEMENT)
 V = (Y.EV.777K) + 1K
 W = X+V
 X = W.A.777K
 FUNCTION RETURN
 ENTRY TO ADD.
R*****ADDITION
 Z = X+Y
 X = Z.A.777K
 FUNCTION RETURN
 END OF FUNCTION
```

```
 EXTERNAL FUNCTION CLOCK.(F,K)

R***** SIMULATE ACTIVATION NEXT COMMAND SIGNAL

 INTEGER F
 STATEMENT LABEL K
 T'O K(F)
 E'N



 EXTERNAL FUNCTION PRINT.(P)

R***** PRINT OUT OCTAL CONTENTS OF REGISTERS

 N'S INTEGER
 PROGRAM COMMON G,C,D,R,F,A
 P=P+1
 P'T RESULT,P,G,C,D,R,F,A
 W'R F.E.17K.AND.G.E.0.AND.C.E.0.AND.D.E.0
     ERROR RETURN
 O'E
     P'T STATE,F
     F'N
 E'L
 V'S RESULT=$1H ,S6,I2,S24,K1,S3,K2,S3,K2,S3,K3,S3,K2,S3,K3*$
 V'S STATE=$1H+,S18,1HK,K2*$
 E'N




 $DATA
 M(0)  = 510K
 M(1)  = 700K
 M(2)  = 171K
 M(3)  = 312K
 M(4)  = 540K
 M(5)  = 520K
 M(6)  = 410K
 M(8)  = 277K
 M(9)  = 504K
 M(56) = 376K
 M(57) = 156K
 M(63) = 400K*
```

COMPUTER SIMULATION, SEQUENCING CHECK

(ALL REGISTER CONTENTS ARE OCTAL)

CLOCK PULSE	STATE	G	C	D	R	F	A
	POWER ON						
1		0	71	71	771	17	771
2	K17	0	00	00	771	17	771
	START						
3		1	00	00	771	17	771
4	K17	1	00	00	771	06	771
5	K06	1	00	01	510	12	771
6	K12	1	10	01	510	05	771
7	K05	1	10	01	510	16	771
8	K16	1	10	01	510	13	000
9	K13	1	01	01	510	06	000
10	K06	1	01	02	070	12	000
11	K12	1	70	02	070	00	000
12	K00	1	70	02	376	10	000
13	K10	1	70	02	376	13	376
14	K13	1	02	02	376	06	376
15	K06	1	02	03	171	12	376
16	K12	1	71	03	171	01	376
17	K01	1	71	03	156	11	376
18	K11	1	71	03	156	13	220
19	K13	1	03	03	156	06	220
20	K06	1	03	04	312	12	220
21	K12	1	12	04	312	03	220
22	K03	1	12	04	312	13	220
23	K13	1	04	04	312	06	220
24	K06	1	04	05	540	12	220
25	K12	1	40	05	540	05	220
26	K05	1	40	05	540	14	220
27	K14	1	40	05	540	13	110
28	K13	1	05	05	540	06	110
29	K06	1	05	06	520	12	110
30	K12	1	20	06	520	05	110
31	K05	1	20	06	520	15	110
32	K15	1	20	06	520	13	220
33	K13	1	06	06	520	06	220
34	K06	1	06	07	410	12	220
35	K12	1	10	07	410	04	220
36	K04	1	10	10	410	13	220
37	K13	1	10	10	410	06	220
38	K06	1	10	11	277	12	220
39	K12	1	77	11	277	02	220
40	K02	1	77	11	277	13	220
41	K13	1	11	11	277	06	220
42	K06	1	11	12	504	12	220
43	K12	1	04	12	504	05	220
44	K05	0	04	12	504	17	220
45	K17	0	00	00	504	17	220
	STOP						

APPENDIX D

TABULATION PROGRAM

```
          N'S INTEGER
          D'N V(3),W(1)
START     L'T IN1,STATE,COND,V(0)...V(3)

         R***** TEST FOR CONDITIONAL STATEMENT

          W'R COND.E.$W'R$.OR.COND.E.$O'R$
              T'O CON

         R***** TEST FOR END OF CONDITIONAL

          O'R COND.E.$E'L$
              R'T IN1
              T'O START

         R***** TEST FOR ILLEGAL CONDITIONAL STATEMENT

          O'R COND.E.$O'E$
              T'O ERROR

         R***** TEST FOR END OF PROGRAM

          O'R COND.E.$END$.OR.COND.E.$E'M$
              T'O FINIS
          O'E
              T'O NOCON
          E'L

         R***** READ CONDITIONAL STATEMENT

CON       R'T IN1,STATE,COND,V(0)...V(3)
          W'R STATE.E.$POWER$.OR.STATE.E.$START$.OR.STATE.E.$      $
              C'E
          O'E

         R***** ELIMINATE K FROM STATEMENT LABEL

              STATE=(STATE.A.77777777K4).V.$0000) $
          E'L

         R***** TEST FOR G AND POSITION FOR COLUMN HEADING

          W'R (V(0).A.77K10).E.$G00000$
              SWV=$  G $
              VAL=V(0).LS.24
```

```
R***** TEST FOR A(0) AND CONDITIONAL JUMP

 O'R (V(0).A.77K8).E.$0A0000$
     SWV=$A(0)$
     VAL=$1$
     W'R V(3).E.$=R.A.7$
         REG=(V(2).LS.30).V.$0      $
         CONT=$R(ADDR$
     O'E
         T'O GO
     E'L
     P'T OUT,STATE,SWV,VAL,REG,CONT
     PUNCH FORMAT OUT,STATE,SWV,VAL,REG,CONT
     T'O START

R***** TEST FOR C-BIT CONDITIONALS

 O'R (V(0).A.77K8).E.$0C0000$
     W'R (V(1).A.77K10).E.$K00000$.AND.(V(0).A.77K).E.4
         SWV=$C(3)$
     O'R (V(1).A.7777K8).E.$0K0000$
         W'R (V(0).A.77K).E.1
             SWV=$C(2)$
         O'R (V(0).A.77K).E.2
             SWV=$C(1)$
         O'R (V(0).A.77K).E.4
             SWV=$C(0)$
         O'E
             T'O ERROR
         E'L
     E'L
     VAL=$1$
 E'L
GO   L'T IN1,SX,CX,V(0)...V(3)

R***** TEST FOR NEXT CONDITIONAL OR COMMAND SIGNAL

 W'R SX.NE.$      $.OR.CX.NE.$    $, T'O START
 R'T IN1,SX,CX,V(0)...V(3)
 REG=(V(0).A.77K10).V.$0      $
 CONT=(V(0).LS.12).V.(V(1).RS.24)

R***** RIGHT ADJUST OCTAL VALUE OF F

 W'R REG.E.$F$.AND.(CONT.A.77K8).E.$0K0000$
     CONT=((CONT.A.77K10).RS.6).V.$00     $
 O'R REG.E.$F$.AND.(CONT.A.77K6).E.$00K000$
     CONT=(CONT.A.7777K8).V.$00     $
 E'L
 P'T OUT,STATE,SWV,VAL,REG,CONT
 PUNCH FORMAT OUT,STATE,SWV,VAL,REG,CONT
```

```
R***** CLEAR 'STATE','SWV','VAL' IF MULTIPLE PARALLEL
R      OPERATIONS*****

 STATE=$      $
 SWV=$     $
 VAL=$ $
 T'O GO

R***** READ UNCONDITIONAL OPERATIONS

NOCON    R'T IN3,STATE,W(0),W(1)
 W'R STATE.E.$POWER$.OR.STATE.E.$START$.OR.STATE.E.$      $
     C'E
 O'E
 STATE=(STATE.A.77777777K4).V.$0000) $
 E'L
 REG=(W(0).A.77K10).V.$0      $
 CONT=(W(0).LS.12).V.(W(1).RS.24)

R***** TEST FOR R(ADDR) TRANSFER TO C OR D

 W'R CONT.E.$R.A.77$, CONT=$R(ADDR$

R***** TEST FOR R(OP) TRANSFER TO F

 W'R REG.E.$F$.AND.CONT.E.$((R.A.$
     P'T F0,STATE
     PUNCH FORMAT F0,STATE
     P'T FI,STATE
     PUNCH FORMAT FI,STATE
     T'O START

R***** RIGHT ADJUST OCTAL VALUE OF F

 O'R REG.E.$F$.AND.(CONT.A.77K8).E.$0K0000$
 CONT=((CONT.A.77K10).RS.6).V.$00     $
 O'R REG.E.$F$.AND.(CONT.A.77K6).E.$00K000$
 CONT=(CONT.A.7777K8).V.$00     $

R***** TEST FOR STORE

 O'R REG.E.$M$
     REG=$M(C)$
     CONT=$A$
 E'L
 P'T OUT2,STATE,REG,CONT
 PUNCH FORMAT OUT2,STATE,REG,CONT
 T'O START
```

```
ERROR     PRINT COMMENT $0                ERROR$
          T'O END

FINIS     P'T FINAL
          PUNCH FORMAT FINAL

          V'S IN1 = $C5,S6,C3,S1,4C6*$
          V'S IN3 = $C5,S6,2C6*$
          V'S OUT = $1H ,C5,S6,C4,S2,C1,S6,C4,S2,C6*$
          V'S OUT2 = $1H ,C5,S19,C4,S2,C6*$
          V'S F0 = $1H ,C5,S19,7HF(0)  0*$
          V'S FI = $1H ,C5,S19,11HF(I)  R(OP)*$
          V'S FINAL = $1H ,S11,3HEND*$
END       E'M
```

```
$DATA
POWER       F=17K
            G=0
START       G=1
K(17K)      W'R G.E.0
                C=0
                D=0
            O'R G.E.1
                F=6K
            E'L
K(06K)      R=M(C)
            D=UCOUNT.(D)
            W'R G.E.0
                F=17K
            O'R G.E.1
                F=12K
            E'L
K(12K)      C=R.A.77K
            F=((R.A.700K).RS.6).A.7K
K(00K)      R=M(C)
            F=10K
K(10K)      A=ADD.(A,R)
            F=13K
K(01K)      R=M(C)
            F=11K
K(11K)      A=SUB.(A,R)
            F=13K
K(04K)      D=R.A.77K
            F=13K
K(02K)      F=13K
            W'R (A.A.400K).NE.0, D=R.A.77K
K(03K)      M(C)=A
            F=13K
K(05K)      W'R (C.A.4K).NE.0
                G=0
                F=17K
            O'R (C.A.10K).NE.0
                F=16K
            O'R (C.A.20K).NE.0
                F=15K
            O'R (C.A.40K).NE.0
                F=14K
            E'L
K(13K)      C=D
            F=6K
K(14K)      A=SHRA.(A)
            F=13K
K(15K)      A=CIRL.(A)
            F=13K
K(16K)      A=0
            F=13K
            END OF PROGRAM
```

APPENDIX E

TABULATION PROGRAM OUTPUT

```
POWER                       F       17
                            G       0
START                       G       1
K(17)       G    0          C       0
                            D       0
            G    1          F       06
K(06)                       R       M(C)
                            D       UCOUNT
            G    0          F       17
            G    1          F       12
K(12)                       C       R(ADDR
                            F(0)    0
                            F(I)    R(OP)
K(00)                       R       M(C)
                            F       10
K(10)                       A       ADD.(A
                            F       13
K(01)                       R       M(C)
                            F       11
K(11)                       A       SUB.(A
                            F       13
K(04)                       D       R(ADDR
                            F       13
K(02)                       F       13
            A(0)    1       D       R(ADDR
K(03)                       M(C)    A
                            F       13
K(05)       C(3)    1       G       0
                            F       17
            C(2)    1       F       16
            C(1)    1       F       15
            C(0)    1       F       14
K(13)                       C       D
                            F       06
K(14)                       A       SHRA.(
                            F       13
K(15)                       A       CIRL.(
                            F       13
K(16)                       A       0
                            F       13
            END
```

APPENDIX F

TRUTH TABLE PROGRAM

```
          N'S INTEGER
          D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
          PROGRAM COMMON STATE,SWV,VAL,REG,CONT
          P'N CLR(2),TRF(3),HEAD2(2)
START     T'H L1, FOR I=1,1, I.G.100
          R'T IN,STATE(I),SWV(I),VAL(I),REG(I),CONT(I)
          W'R SWV(I).E.$END $, T'O EXIT

         R***** REPEAT 'STATE','SWV',AND'VAL', WHERE APPLICABLE,
         R      FOR ALL TRANSFER OPERATIONS*****

          W'R STATE(I).E.$      $.AND.SWV(I).E.$    $, SWV(I)=SWV(I-1)
          W'R STATE(I).E.$      $.AND.VAL(I).E.$ $, VAL(I)=VAL(I-1)
          W'R STATE(I).E.$      $, STATE(I)=STATE(I-1)
L1        C'E
EXIT      PRINT COMMENT $1      F-REGISTER TRUTH TABLES$
          FTAB.(I)
          PRINT COMMENT $1      G-REGISTER TRUTH TABLES$
          GTAB.(I)
          PRINT COMMENT $1      C-REGISTER TRUTH TABLES$
          CTAB.(I)
          PRINT COMMENT $1      D-REGISTER TRUTH TABLES$
          DTAB.(I)
          PRINT COMMENT $1      R-REGISTER TRUTH TABLES$
          RTAB.(I)
          PRINT COMMENT $1      MEMORY TRUTH TABLES$
          MEMTAB.(I)
          PRINT COMMENT $1      A-REGISTER TRUTH TABLES$
          ATAB.(I)
          V'S IN = $S1,C5,S6,C4,S2,C1,S6,C4,S2,C6*$
          V'S HEAD2 = $1H0,C5,S25,C5*$
          V'S CLR = $1H0,S15,C4,S31,C4*$
          V'S TRF = $1H0,S5,C6,S4,C6,S29,C6*$
          E'M
```

```
 EXTERNAL FUNCTION FTAB.(I)

R***** GENERATE ONE LINE F-REGISTER TRUTH TABLES

 N'S INTEGER
 PROGRAM COMMON STATE,SWV,VAL,REG,CONT
 D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
 D'N X(3),Y(3),Z(3)
 T'H L2, FOR J=1,1, J.G.I-1

R***** TEST FOR END OF F OPERATIONS

 W'R REG(J).NE.$F    $.AND.REG(J).NE.$F(I)$, T'O L2
 W'R STATE(J).E.$POWER$.OR.STATE(J).E.$START$, T'O L2

R***** CONVERT STARTING OCTAL VALUE OF F TO BINARY

 X(0)=(STATE(J).A.1K6).RS.18
 X(1)=(STATE(J).A.4K4).RS.14
 X(2)=(STATE(J).A.2K4).RS.13
 X(3)=(STATE(J).A.1K4).RS.12

R***** TEST FOR R(OP) TRANSFER TO F

 W'R REG(J).E.$F(I)$
      T'H L3, FOR K=0,1, K.G.5
      ROP=K

R.    ***** SIMULATE 3-BIT R(OP) TRANSFER TO F

      Y(1)=(ROP.A.4K).RS.2
      Y(2)=(ROP.A.2K).RS.1
      Y(3)=ROP.A.1K

R     ***** F(0) SET TO 0

      ZZ=0
      PUNCH FORMAT HEAD,STATE(J)
      P'T HEAD,STATE(J)
      PUNCH FORMAT OUT1,X(0)...X(3),Y(1)...Y(3),ZZ,Y(1)...Y(3)
      P'T OUT1,X(0)...X(3),Y(1)...Y(3),ZZ,Y(1)...Y(3)
L3    C'E
 O'R REG(J).E.$F    $

R     ***** READ VALUE OF CONDITIONAL VARIABLE, IF ANY

      YY=VAL(J)
```

```
R     ***** CONVERT FINAL OCTAL VALUE OF F TO BINARY

      Z(0)=(CONT(J).A.1K10).RS.30
      Z(1)=(CONT(J).A.4K8).RS.26
      Z(2)=(CONT(J).A.2K8).RS.25
      Z(3)=(CONT(J).A.1K8).RS.24
      PUNCH FORMAT HEAD1,STATE(J),SWV(J)
      P'T HEAD1,STATE(J),SWV(J)
      PUNCH FORMAT OUT2,X(0)...X(3),YY,Z(0)...Z(3)
      P'T OUT2,X(0)...X(3),YY,Z(0)...Z(3)
 E'L
L2    C'E
 F'N
 V'S HEAD=$1H0,C5,S5,34HF(0) F(1) F(2) F(3) R(0) R(1) R(2),
1S6,19HF(0) F(1) F(2) F(3)*$
 V'S HEAD1=$1H0,C5,S5,19HF(0) F(1) F(2) F(3),S1,C4,S16,
119HF(0) F(1) F(2) F(3)*$
 V'S OUT1=$1H ,S10,7(S2,I1,S2),S5,4(S2,I1,S2)*$
 V'S OUT2=$1H ,S10,4(S2,I1,S2),S2,C1,S17,4(S2,I1,S2)*$
 E'N
```

```
      EXTERNAL FUNCTION GTAB.(I)
      N'S INTEGER
      PROGRAM COMMON STATE,SWV,VAL,REG,CONT
      D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
      P'N CLR(2),TRF(3),HEAD2(2)
      GBIT = $  G $
      T'H L4, FOR L=1,1, L.G.I-1
      W'R SWV(L).E.$  G $, SWV(L)=$   G$

R***** COMPLEMENT ALL CONDITIONAL VARIABLES IF TRUTH VALUE 0

      W'R VAL(L).E.$0$, SWV(L)=(SWV(L).A.77777777K4).V.$0000' $

R***** TEST FOR END OF G OPERATIONS

      W'R REG(L).NE.$G   $, T'O L4

R***** PRINT COMMAND SIGNAL AND CONDITIONAL VARIABLE

      PUNCH FORMAT HEAD2,STATE(L),SWV(L)
      P'T HEAD2,STATE(L),SWV(L)

R***** PRINT TRUTH TABLE HEADINGS

      PUNCH FORMAT CLR,GBIT,GBIT
      P'T CLR,GBIT,GBIT

R***** BB IS FINAL VALUE OF G

      .
      BB=0
      W'R CONT(L).E.$1     $, BB=1
      PRNT1.(BB)
L4    C'E
      F'N
      E'N
```

```
      EXTERNAL FUNCTION CTAB.(I)
      N'S INTEGER
      PROGRAM COMMON STATE,SWV,VAL,REG,CONT
      D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
      P'N CLR(2),TRF(3),HEAD2(2)

     R***** DEFINE COLUMN HEADINGS

      V'S CBITS = $C(I)$,$D(I)$,$R(I+3)$
      T'H L6, FOR N=1,1, N.G.I-1

     R***** TEST FOR END OF C OPERATIONS

      W'R REG(N).NE.$C    $, T'O L6

     R***** PRINT COMMAND SIGNAL AND CONDITIONAL VARIABLE

      PUNCH FORMAT HEAD2,STATE(N),SWV(N)
      P'T HEAD2,STATE(N),SWV(N)

     R***** TEST FOR CLEAR

      W'R CONT(N).E.$0      $
          PUNCH FORMAT CLR,CBITS(0),CBITS(0)
          P'T CLR,CBITS(0),CBITS(0)
          BB=0
          PRNT1.(BB)
      O'E
      .   W'R CONT(N).E.$D      $
          PUNCH FORMAT TRF,CBITS(0),CBITS(1),CBITS(0)
          P'T TRF,CBITS(0),CBITS(1),CBITS(0)
          O'R CONT(N).E.$R(ADDR$
          PUNCH FORMAT TRF,CBITS(0),CBITS(2),CBITS(0)
          P'T TRF,CBITS(0),CBITS(2),CBITS(0)
          O'E
              T'O L6
          E'L
          PRNT2.(AZ)
      E'L
L6    C'E
      F'N
      E'N
```

```
      EXTERNAL FUNCTION DTAB.(I)
      N'S INTEGER
      PROGRAM COMMON STATE,SWV,VAL,REG,CONT
      D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
      P'N CLR(2),TRF(3),HEAD2(2)

     R***** DEFINE COLUMN HEADINGS

      V'S DBITS = $D(I)$,$R(I+3)$
      T'H L9, FOR Q=1,1, Q.G.I-1

     R***** TEST FOR END OF D OPERATIONS

      W'R REG(Q).NE.$D    $, T'O L9

     R***** PRINT COMMAND SIGNAL AND CONDITIONAL VARIABLE

      PUNCH FORMAT HEAD2,STATE(Q),SWV(Q)
      P'T HEAD2,STATE(Q),SWV(Q)

     R***** TEST FOR COUNT

      W'R CONT(Q).E.$UCOUNT$
          PUNCH FORMAT HEADD1
          P'T HEADD1
          PCNT.(AZ)

     R***** TEST FOR R(ADDR) TO D

      O'R CONT(Q).E.$R(ADDR$
          PUNCH FORMAT TRF,DBITS(0),DBITS(1),DBITS(0)
          P'T TRF,DBITS(0),DBITS(1),DBITS(0)
          PRNT2.(AZ)

     R***** TEST FOR CLEAR

      O'R CONT(Q).E.$0      $
          PUNCH FORMAT CLR,DBITS(0),DBITS(0)
          P'T CLR,DBITS(0),DBITS(0)
          BB=0
          PRNT1.(BB)
      O'E
          C'E
      F'L
L9    C'E
      F'N
      V'S HEADD1 = $1H0,29HD(0) D(1) D(2) D(3) D(4) D(5),S21,
     129HD(0) D(1) D(2) D(3) D(4) D(5)*$
      E'N
```

```
      EXTERNAL FUNCTION RTAB.(I)
      N'S INTEGER
      PROGRAM COMMON STATE,SWV,VAL,REG,CONT
      D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
      P'N CLR(2),TRF(3),HEAD2(2)

     R***** DEFINE COLUMN HEADINGS

      V'S RBITS = $R(I)$,$M(C,I)$
      T'H L13, FOR NN=1,1, NN.G.I-1

     R***** TEST FOR END OF R OPERATIONS

      W'R REG(NN).NE.$R    $, T'O L13

     R***** PRINT COMMAND SIGNAL AND CONDITIONAL VARIABLE

      PUNCH FORMAT HEAD2,STATE(NN),SWV(NN)
      P'T HEAD2, STATE(NN),SWV(NN)
      PUNCH FORMAT TRF,RBITS(0),RBITS(1),RBITS(0)
      P'T TRF,RBITS(0),RBITS(1),RBITS(0)
      PRNT2.(AZ)
L13   C'E
      F'N
      E'N



      EXTERNAL FUNCTION MEMTAB.(I)
      N'S INTEGER
      PROGRAM COMMON STATE,SWV,VAL,REG,CONT
      D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
      P'N CLR(2),TRF(3),HEAD2(2)

     R***** DEFINE COLUMN HEADINGS

      V'S MBITS = $M(C,I)$,$A(I)$
      T'H L15, FOR QQ=1,1, QQ.G.I-1

     R***** TEST FOR END OF M OPERATIONS

      W'R REG(QQ).NE.$M(C)$, T'O L15

     R***** PRINT COMMAND SIGNAL AND CONDITIONAL VARIABLE

      PUNCH FORMAT HEAD2,STATE(QQ),SWV(QQ)
      P'T HEAD2,STATE(QQ),SWV(QQ)
      PUNCH FORMAT TRF,MBITS(0),MBITS(1),MBITS(0)
      P'T TRF,MBITS(0),MBITS(1),MBITS(0)
      PRNT2.(AZ)
L15   C'E
      F'N
      E'N
```

```
 EXTERNAL FUNCTION ATAB.(I)
 N'S INTEGER
 PROGRAM COMMON STATE,SWV,VAL,REG,CONT
 D'N STATE(100),SWV(100),VAL(100),REG(100),CONT(100)
 P'N CLR(2),TRF(3),HEAD2(2)

R***** DEFINE COLUMN HEADINGS

 V'S ABITS = $A(0)$,$A(1)$,$A(I)$,$A(I+1)$,$A(I-1)$,$A(7)$,
1$A(8)$,$R(0)$,$R(I)$,$R(8)$,$B(0)$,$B(I)$,$B(8)$,$B(I-1)$,
2$B(7)$

R***** DEFINE SWITCHES FOR A(0) TABLE AND ADD/SUB

 BOOLEAN AX,AY
 T'H L16, FOR II=1,1, II.G.I-1

R***** TEST FOR END OF A OPERATIONS

 W'R REG(II).NE.$A   $, T'O L16
 PUNCH FORMAT HEAD2,STATE(II),SWV(II)
 P'T HEAD2,STATE(II),SWV(II)

R***** TEST FOR CLEAR

 W'R CONT(II).E.$0     $
      BB=0
      PUNCH FORMAT CLR,ABITS(0),ABITS(0)
      P'T CLR,ABITS(0),ABITS(0)
      PRNT1.(BB)
      PUNCH FORMAT CLR,ABITS(2),ABITS(2)
      P'T CLR,ABITS(2),ABITS(2)
      PRNT1.(BB)
      PUNCH FORMAT CLR,ABITS(6),ABITS(6)
      P'T CLR,ABITS(6),ABITS(6)
      PRNT1.(BB)

R***** TEST FOR SHRA

 O'R CONT(II).E.$SHRA.($
      PUNCH FORMAT TRF,ABITS(2),ABITS(4),ABITS(2)
      P'T TRF,ABITS(2),ABITS(4),ABITS(2)
      PRNT2.(AZ)
      PUNCH FORMAT TRF,ABITS(6),ABITS(5),ABITS(6)
      P'T TRF,ABITS(6),ABITS(5),ABITS(6)
      PRNT2.(AZ)
```

```
R***** TEST FOR CIRL

 O'R CONT(II).E.$CIRL.($
      PUNCH FORMAT TRF,ABITS(0),ABITS(1),ABITS(0)
      P'T TRF,ABITS(0),ABITS(1),ABITS(0)
      PRNT2.(AZ)
      PUNCH FORMAT TRF,ABITS(2),ABITS(3),ABITS(2)
      P'T TRF,ABITS(2),ABITS(3),ABITS(2)
      PRNT2.(AZ)
      PUNCH FORMAT TRF,ABITS(6),ABITS(0),ABITS(6)
      P'T TRF,ABITS(6),ABITS(0),ABITS(6)
      PRNT2.(AZ)
R***** TEST FOR ADD/SUB
 O'R CONT(II).E.$ADD.(A$.OR.CONT(II).E.$SUB.(A$

R     ***** A(0) TABLE

      AY=1B

R     ***** ADD

      AX=1B

R     ***** SUB

      W'R CONT(II).E.$SUB.(A$, AX=0B
      PUNCH FORMAT ADSUB,ABITS(0),ABITS(7),ABITS(10),ABITS(0)
      P'T  ADSUB,ABITS(0),ABITS(7),ABITS(10),ABITS(0)
      PAS.(AX,AY)

R     ***** OTHER A-BIT TABLES

      AY=0B
      PUNCH FORMAT ADSUB,ABITS(2),ABITS(8),ABITS(11),ABITS(2),
1     ABITS(13)
      P'T  ADSUB,ABITS(2),ABITS(8),ABITS(11),ABITS(2),ABITS(13)
      PAS.(AX,AY)
      PUNCH FORMAT ADSUB,ABITS(6),ABITS(9),ABITS(12),ABITS(6),
1     ABITS(14)
      P'T  ADSUB,ABITS(6),ABITS(9),ABITS(12),ABITS(6),ABITS(14)
      PAS.(AX,AY)
 E'L
L16   C'E
 F'N
 V'S ADSUB = $1H0,S5,C6,S4,C6,S4,C6,S19,C6,S4,C6*$
 E'N
```

```
            EXTERNAL FUNCTION (X)
            N'S INTEGER

           R***** GENERATE ONE VARIABLE TRUTH TABLE (CLEAR OR SET G TO 1)

            E'O PRNT1.
            T'H LP1, FOR I=0,1, I.G.1
            A=I
            B=X
            PUNCH FORMAT CLR,A,B
LP1         P'T CLR,A,B
            F'N

           R***** GENERATE TWO VARIABLE TRUTH TABLE (TRANSFER)

            E'O PRNT2.
            T'H LP2, FOR J=0,1, J.G.3
            AT=J.A.1K
            BT=(J.A.2K).RS.1
            PUNCH FORMAT TRF,BT,AT,AT
LP2         P'T TRF,BT,AT,AT
            F'N
            V'S CLR = $1H ,S17,I1,S34,I1*$
            V'S TRF = $1H ,S7,I1,S9,I1,S34,I1*$
            E'N



            EXTERNAL FUNCTION PCNT.(Y)

           R***** GENERATE SIX BIT COUNTER TRUTH TABLE

            N'S INTEGER
            D'N X(11)
            T'H LOOP, FOR I=0,1, I.G.63
            K=I
            L=I+1
            T'H LIP, FOR J=0,1, J.G.5
                X(5-J)=K.A.1K
                X(11-J)=L.A.1K
                K=K.RS.1
LIP             L=L.RS.1
            PUNCH FORMAT CNT,X(0)...X(11)
LOOP        P'T CNT,X(0)...X(11)
            F'N
            V'S CNT = $1H ,6(S2,I1,S2),S20,6(S2,I1,S2)*$
            E'N
```

```
          EXTERNAL FUNCTION PAS.(X,Y)

         R***** GENERATE ADD/SUB TRUTH TABLES

          N'S INTEGER
          BOOLEAN X,Y
          T'H LOOP, FOR I=0,1, I.G.7
          A=(I.A.4K).RS.2
          B=(I.A.2K).RS.1
          C=I.A.1K
          W'R X

         R***** ADD

               F=A+B+C
          O'E

         R***** SUB

               F=A+((.N.B).A.1K)+C
          E'L
          D=F.A.1K
          E=(F.A.2K).RS.1
          W'R Y

         R***** A(0) TABLE, SUPPRESS CARRY(B)

               PUNCH FORMAT OUTA,A,B,C,D
               P'T OUTA,A,B,C,D
          O'E

         R***** A(I) AND A(8) TABLE, GENERATE CARRY(B)

               PUNCH FORMAT OUTA,A,B,C,D,E
               P'T OUTA,A,B,C,D,E
          E'L
LOOP      C'E
OUT       F'N
          V'S OUTA = $1H ,S7,I1,S9,I1,S9,I1,S24,I1,S9,I1*$
          E'N
```

APPENDIX G

TRUTH TABLE PROGRAM OUTPUT

```
OK(17)      F(0) F(1) F(2) F(3)   G
              1    1    1    1    1
OK(06)      F(0) F(1) F(2) F(3)   G
              0    1    1    0    0
OK(06)      F(0) F(1) F(2) F(3)   G
              0    1    1    0    1
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    0    0    0
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    0    0    1
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    0    1    0
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    0    1    1
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    1    0    0
OK(12)      F(0) F(1) F(2) F(3) R(0) R(1) R(2)
              1    0    1    0    1    0    1
OK(00)      F(0) F(1) F(2) F(3)
              0    0    0    0
OK(10)      F(0) F(1) F(2) F(3)
              1    0    0    0
OK(01)      F(0) F(1) F(2) F(3)
              0    0    0    1
OK(11)      F(0) F(1) F(2) F(3)
              1    0    0    1
OK(04)      F(0) F(1) F(2) F(3)
              0    1    0    0
OK(02)      F(0) F(1) F(2) F(3)
              0    0    1    0
OK(03)      F(0) F(1) F(2) F(3)
              0    0    1    1
OK(05)      F(0) F(1) F(2) F(3) C(3)
              0    1    0    1    1
OK(05)      F(0) F(1) F(2) F(3) C(2)
              0    1    0    1    1
OK(05)      F(0) F(1) F(2) F(3) C(1)
              0    1    0    1    1
OK(05)      F(0) F(1) F(2) F(3) C(0)
              0    1    0    1    1
OK(13)      F(0) F(1) F(2) F(3)
              1    0    1    1
OK(14)      F(0) F(1) F(2) F(3)
              1    1    0    0
OK(15)      F(0) F(1) F(2) F(3)
              1    1    0    1
OK(16)      F(0) F(1) F(2) F(3)
              1    1    1    0
            F(0) F(1) F(2) F(3)
              0    1    1    0
            F(0) F(1) F(2) F(3)
              1    1    1    1
            F(0) F(1) F(2) F(3)
              1    0    1    0
            F(0) F(1) F(2) F(3)
              0    0    0    0
            F(0) F(1) F(2) F(3)
              0    0    0    1
            F(0) F(1) F(2) F(3)
              0    0    1    0
            F(0) F(1) F(2) F(3)
              0    0    1    1
            F(0) F(1) F(2) F(3)
              0    1    0    0
            F(0) F(1) F(2) F(3)
              0    1    0    1
            F(0) F(1) F(2) F(3)
              1    0    0    0
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    0    0    1
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    1    1    1
            F(0) F(1) F(2) F(3)
              1    1    1    0
            F(0) F(1) F(2) F(3)
              1    1    0    1
            F(0) F(1) F(2) F(3)
              1    1    0    0
            F(0) F(1) F(2) F(3)
              0    1    1    0
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    0    1    1
            F(0) F(1) F(2) F(3)
              1    0    1    1
```

```
OPOWER
0
                    G
                    0
                    1
OSTART
0
                    G
                    0
                    1
OK(05)
0
                    G
                    0
                    1
OK(17)
0
                  C(I)
                    0
                    1
OK(12)
0       C(I)      R(I+3)
          0         0
          0         1
          1         0
          1         1
OK(13)
0       C(I)      D(I)
          0         0
          0         1
          1         0
          1         1
OK(17)
0                   D(I)
                      0
                      1
OK(04)
0       D(I)      R(I+3)
          0         0
          0         1
          1         0
          1         1
OK(02)
0       D(I)      R(I+3)
          0         0
          0         1
          1         0
          1         1

                                    G
                                    0
                                    0

                                    G
                                    1
                                    1
                  C(3)
                                    G
                                    0
                                    0
                     G'
                                  C(I)
                                    0
                                    0

                                  C(I)
                                    0
                                    1
                                    0
                                    1

                                  C(I)
                                    0
                                    1
                                    0
                                    1
                     G'
                                  D(I)
                                    0
                                    0

                                  D(I)
                                    0
                                    1
                                    0
                                    1
                  A(0)
                                  D(I)
                                    0
                                    1
                                    0
                                    1
```

OK(06)

D(0)	D(1)	D(2)	D(3)	D(4)	D(5)
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	0	1
0	1	0	1	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	0	1
0	1	1	1	1	0
0	1	1	1	1	1

D(0)	D(1)	D(2)	D(3)	D(4)	D(5)
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	0	1
0	1	0	1	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	0	1
0	1	1	1	1	0
0	1	1	1	1	1
1	0	0	0	0	0

1	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	0	1	1
1	0	1	1	0	0
1	0	1	1	0	1
1	0	1	1	1	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	0	1
1	1	0	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	0	1
1	1	1	1	1	0
1	1	1	1	1	1

1	0	0	0	0	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	0	1	1
1	0	1	1	0	0
1	0	1	1	0	1
1	0	1	1	1	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	0	1
1	1	0	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	0	1
1	1	1	1	1	0
1	1	1	1	1	1

0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	0	1
0	1	0	1	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	0	1
0	1	1	1	1	0
0	1	1	1	1	1

0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	0	1
0	1	0	0	1	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	0	1
0	1	0	1	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	0	1
0	1	1	1	1	0
0	1	1	1	1	1

```
OK(06)
0       R(I)        M(C,I)
          0           0
          0           1
          1           0
          1           1
                                  R(I)
                                    0
                                    1
                                    0
                                    1
OK(00)
0       R(I)        M(C,I)
          0           0
          0           1
          1           0
          1           1
                                  R(I)
                                    0
                                    1
                                    0
                                    1
OK(01)
0       R(I)        M(C,I)
          0           0
          0           1
          1           0
          1           1
                                  R(I)
                                    0
                                    1
                                    0
                                    1
OK(03)
0       M(C,I)      A(I)
          0           0
          0           1
          1           0
          1           1
                                  M(C,I)
                                    0
                                    1
                                    0
                                    1
```

```
OK(10)
0       A(0)        R(0)        B(0)        A(0)
          0           0           0           0
          0           0           1           1
          0           1           0           1
          0           1           1           0
          1           0           0           1
          1           0           1           0
          1           1           0           0
          1           1           1           1
0       A(I)        R(I)        B(I)        A(I)        B(I-1)
          0           0           0           0           0
          0           0           1           1           0
          0           1           0           1           0
          0           1           1           0           1
          1           0           0           1           0
          1           0           1           0           1
          1           1           0           0           1
          1           1           1           1           1
0       A(8)        R(8)        B(8)        A(8)        B(7)
          0           0           0           0           0
          0           0           1           1           0
          0           1           0           1           0
          0           1           1           0           1
          1           0           0           1           0
          1           0           1           0           1
          1           1           0           0           1
          1           1           1           1           1
OK(11)
0       A(0)        R(0)        B(0)        A(0)
          0           0           0           1
          0           0           1           0
          0           1           0           0
          0           1           1           1
          1           0           0           0
          1           0           1           1
          1           1           0           1
          1           1           1           0
0       A(I)        R(I)        B(I)        A(I)        B(I-1)
          0           0           0           1           0
          0           0           1           0           1
          0           1           0           0           0
          0           1           1           1           0
          1           0           0           0           1
          1           0           1           1           1
          1           1           0           1           0
          1           1           1           0           1
```

```
0       A(8)        R(8)        B(8)
          0           0           0
          0           0           1
          0           1           0
          0           1           1
          1           0           0
          1           0           1
          1           1           0
          1           1           1
OK(14)
0       A(I)        A(I-1)
          0           0
          0           1
          1           0
          1           1
0       A(8)        A(7)
          0           0
          0           1
          1           0
          1           1
OK(15)
0       A(0)        A(1)
          0           0
          0           1
          1           0
          1           1
0       A(I)        A(I+1)
          0           0
          0           1
          1           0
          1           1
0       A(8)        A(0)
          0           0
          0           1
          1           0
          1           1
OK(16)
0                   A(0)
                      0
                      1
0                   A(I)
                      0
                      1
0                   A(8)
                      0
                      1
```

```
A(8)        B(7)
  1           0
  0           1
  0           0
  1           0
  0           1
  1           1
  1           0
  0           1

A(I)
  0
  1
  0
  1
A(8)
  0
  1
  0
  1

A(0)
  0
  1
  0
  1
A(I)
  0
  1
  0
  1
A(8)
  0
  1
  0
  1

A(0)
  0
  0
A(I)
  0
  0
A(8)
  0
  0
```

APPENDIX H

BOOLEAN EXPRESSION PROGRAM

```
          N'S INTEGER
          D'N REG(3),SWV(2),F(3),V(2),FN(3),B(4),DX(11)

         R***** SET UP J- AND K- SUFFIX

          V'S FLF = $K$,$J$
          P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
          P'N F(3),V(2),FN(3)
          T'H LP0, FOR L=0,1, L.G.100
          L'T IN1,STATE,REG(0)...REG(3),SWV(0)...SWV(2)

         R***** TEST FOR END OF F TRUTH TABLES

          W'R REG(0).NE.$F(0)$, T'O NEXT
          R'T IN1,STATE,REG(0)...REG(3),SWV(0)...SWV(2)
          R'T IN2,F(0)...F(3),V(0)...V(2),FN(0)...FN(3)
          VBLF.(NX)
LP0       C'E
NEXT      R'T STS,STATE,SWV(0)
          L'T DREG,DX(0)...DX(11)

         R***** TEST FOR COUNTER TRUTH TABLE

          W'R DX(2).E.$D(2)$, T'O DCOUNT
A3        R'T RGSTR,B(0)...B(4)

         R***** TEST FOR ONE OR TWO VARIABLE TRUTH TABLE

          W'R B(2).E.$       $
              W'R B(0).E.$       $
                  VBL1.(NX)
              O'E

         R***** ADD/SUB TRUTH TABLE

                  VBL2.(NX)
              E'L
          O'E
              VBL32.(NX)
          E'L
```

```
          W'R (B(1).E.$A(0)$.OR.B(1).E.$A(I)$).AND.B(0).E.$        $

         R***** READ A(I) AND A(8) TRUTH TABLES

              T'O A3
          O'R B(0).E.$A(0)$.OR.B(0).E.$A(I)$
              T'O A3
          O'E
              T'O NEXT
          E'L
DCOUNT    R'T DREG,DX(0)...DX(11)

         R***** COUNTER TRUTH TABLE

          PCNTR.(NX)
          T'O NEXT
          V'S IN1 = $S1,C5,S5,7(C4,S1)*$
          V'S IN2 = $S13,7(C1,S4),S5,4(C1,S4)*$
          V'S STS = $S1,C5,S25,C5*$
          V'S RGSTR = $S2,3(S4,C6),S15,2(S4,C6)*$
          V'S DREG = $S1,6(C4,S1),S20,6(C4,S1)*$
          E'M
```

```
      EXTERNAL FUNCTION  VBLF.(NX)

     R***** GENERATE F-REGISTER BOOLEAN TERMS

      N'S INTEGER
      P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
      P'N F(3),V(2),FN(3)
      T'H LP1, FOR I=0,1, I.G.2
          W'R V(I).E.$0$

     R***** COMPLEMENT SWV ON 0

              SWV(I)=(SWV(I).A.77777777K4).V.$0000' $
          O'R V(I).E.$1$
              SWV(I)=(SWV(I).A.77777777K4).V.$0000  $
           O'E
              SWV(I)=$       $
LP1       E'L

     R***** TEST FOR R(OP) TRANSFER TO F(I)

      W'R STATE.E.$K(12)$
          P'T OUT0,REG(0),FLF(0),STATE
          PUNCH FORMAT OUT0,REG(0),FLF(0),STATE
          L=1
      O'E
          L=0
      E'L
      T'H LP2, FOR J=L,1, J.G.3
      W'R F(J).E.$0$.AND.FN(J).E.$1$
          BIT=FLF(1)
      O'R F(J).E.$1$.AND.FN(J).E.$0$
          BIT=FLF(0)
      O'E
          T'O LP2
      E'L

     R***** COMPRESS IF NO CONDITIONAL VARIABLE

      W'R SWV(0).E.$       $
          P'T OUT0,REG(J),BIT,STATE
          PUNCH FORMAT OUT0,REG(J),BIT,STATE

     R***** GENERATE SPECIAL TERMS FOR R(OP) TO F(I) TRANSFER

      O'R SWV(0).E.$R(0)  $.OR.SWV(0).E.$R(0)' $
          P'T OUT,REG(J),BIT,STATE,SWV(J-1)
          PUNCH FORMAT OUT,REG(J),BIT,STATE,SWV(J-1)
```

```
     O'R SWV(0).E.$  G   $

    R***** COMPRESS FURTHER IF CONDITIONAL VARIABLE IS G

          GV=$G     $
          P'T OUT,REG(J),BIT,STATE,GV
          PUNCH FORMAT OUT,REG(J),BIT,STATE,GV
     O'R SWV(0).E.$  G ' $
          GV=$G'    $
          P'T OUT,REG(J),BIT,STATE,GV
          PUNCH FORMAT OUT,REG(J),BIT,STATE,GV
     O'E
          P'T OUT,REG(J),BIT,STATE,SWV(0)
          PUNCH FORMAT OUT,REG(J),BIT,STATE,SWV(0)
     E'L
LP2  C'E
     F'N
     V'S OUT0 = $1H0,C4,C1,3H = ,C5,2H*P*$
     V'S OUT = $1H0,C4,C1,3H = ,C5,3H*P*,C6*$
     E'N
```

```
 EXTERNAL FUNCTION VBL1.(NX)

R***** GENERATE BOOLEAN TERMS FROM ONE VARIABLE TRUTH TABLE

 N'S INTEGER
 P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
 D'N CB(4)
 T'H LOOP, FOR I=0,1, I.G.1
 R'T VAL1,X,Y
 W'R X.E.$0$.AND.Y.E.$1$

R***** TEST FOR J TERM AND COMPLEMENT VARIABLE

      U=$'$
      BIT=FLF(1)
 O'R X.E.$1$.AND.Y.E.$0$

R***** TEST FOR K TERM

      U=$ $
      BIT=FLF(0)
 O'E
      T'O LOOP
 E'L
 W'R B(1).E.$C(I)$.OR.B(1).E.$D(I)$.OR.B(1).E.$A(I)$

R***** GENERATE TERMS FOR EACH BIT OF C,D, AND A

      KS=0
      W'R B(1).E.$A(I)$, KS=1
      KL=5
      W'R B(1).E.$A(I)$, KL=7
      MASK=777700777777K
      T'H LP1, FOR J=KS,1, J.G.KL
      NRC=J.LS.18
      CB(1)=(B(1).A.MASK).V.NRC
      W'R SWV(0).E.$     $

R***** COMPRESS IF NO CONDITIONAL VARIABLE

          P'T OUT1,CB(1),BIT,STATE,CB(1),U
          PUNCH FORMAT OUT1,CB(1),BIT,STATE,CB(1),U
      O'E
          P'T OUT,CB(1),BIT,STATE,SWV(0),CB(1),U
          PUNCH FORMAT OUT,CB(1),BIT,STATE,SWV(0),CB(1),U
      E'L
LP1   C'E
```

```
          O'E
              W'R SWV(0),E.$     $
                  P'T OUT1,B(1),BIT,STATE,B(1),U
                  PUNCH FORMAT OUT1,B(1),BIT,STATE,B(1),U
              O'E
                  P'T OUT,B(1),BIT,STATE,SWV(0),B(1),U
                  PUNCH FORMAT OUT,B(1),BIT,STATE,SWV(0),B(1),U
              E'L
          E'L
LOOP      C'E
          F'N
          V'S VAL1 = $S18,C1,S34,C1*$
          V'S OUT1 = $1H0,C4,C1,3H = ,C5,3H*P*,C4,C1*$
          V'S OUT = $1H0,C4,C1,3H = ,C5,3H*P*,C5,1H*,C4,C1*$
          E'N



          EXTERNAL FUNCTION PCNTR.(NX)

         R***** GENERATE SPECIAL TERMS FOR COUNTER OPERATION

          N'S INTEGER
          P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
          D'N V(5)
          T'H LOOP, FOR I=0,1, I.G.5
LOOP      V(I)=(DX(I).RS.6).V.$*00000$
          T'H LIP, FOR J=0,1, J.G.5
          P'T OUTJ,DX(J),FLF(1),STATE,V(J)...V(5)
          PUNCH FORMAT OUTJ,DX(J),FLF(1),STATE,V(J)...V(5)
          P'T OUTK,DX(J),FLF(0),STATE,V(J)...V(5)
LIP       PUNCH FORMAT OUTK,DX(J),FLF(0),STATE,V(J)...V(5)
          F'N
          V'S OUTJ = $1H0,C4,C1,3H = ,C5,2H*P,C5,1H',5C5*$
          V'S OUTK = $1H0,C4,C1,3H = ,C5,2H*P,6C5*$
          E'N
```

```
 EXTERNAL FUNCTION VBL2.(NX)

R***** GENERATE BOOLEAN FROM TWO VARIABLE TRUTH TABLE

 N'S INTEGER
 P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
 D'N DB(4)
 T'H LOOP, FOR I=0,1, I.G.3
 R'T VAL2,X,Y,Z
 V=$ $

R***** COMPLEMENT SECOND VARIABLE IF TRUTH VALUE IS 0

 W'R Y.E.$0$, V=$'$
 W'R X.E.$0$.AND.Z.E.$1$

R***** TEST FOR J TERM AND COMPLEMENT FIRST VARIABLE

      U=$'$
      BIT=FLF(1)
 O'R X.E.$1$.AND.Y.E.$0$

R***** TEST FOR K TERM
      U=$ $
      BIT=FLF(0)
 O'E
      T'O LOOP
 E'L

R***** TEST FOR M(C,I)

 W'R (B(0).A.7777K).NE.$0000  $
      W'R SWV(0).E.$     $
           P'T OUTU6,B(0),BIT,STATE,B(0),U,B(1),V
           PUNCH FORMAT OUTU6,B(0),BIT,STATE,B(0),U,B(1),V
      O'E
           P'T OUT6,B(0),BIT,STATE,SWV(0),B(0),U,B(1),V
           PUNCH FORMAT OUT6,B(0),BIT,STATE,SWV(0),B(0),U,B(1),V
      E'L
 O'R B(0).E.$C(I)$.OR.B(0).E.$D(I)$.OR.B(0).E.$A(I)$

R***** GENERATE TERMS FOR ALL BITS OF C,D, AND A

      KS=0
      W'R B(0).E.$A(I)$, KS=1
      KL=5
      W'R B(0).E.$A(I)$, KL=7
      MASK1=777700777777K
      MASK2=7777K2
      MASK3=7777K8
```

```
          T'H LP1, FOR J=KS,1, J.G.KL
          NRB=J.LS.18
          NRB1=((J-1).LS.18).V.$000)   $
          NRB2=((J+1).LS.18).V.$000)   $
          NRB3=((J+3).LS.18).V.$000)   $
          DB(0)=(B(0).A.MASK1).V.NRB

        R***** TEST FOR R(I+3),A(I+1),A(I-1)

          W'R (B(1).A.MASK2).E.$000+30$
              DB(1)=(B(1).A.MASK3).V.NRB3
          O'R (B(1).A.MASK2).E.$000+10$
              DB(1)=(B(1).A.MASK3).V.NRB2
          O'R (B(1).A.MASK2).E.$000-10$
              DB(1)=(B(1).A.MASK3).V.NRB1
          O'E
              DB(1)=(B(1).A.MASK1).V.NRB
          E'L
          W'R SWV(0).E.$     $

        R***** COMPRESS IF NO CONDITIONAL VARIABLES

              P'T OUTU4,DB(0),BIT,STATE,DB(0),U,DB(1),V
              PUNCH FORMAT OUTU4,DB(0),BIT,STATE,DB(0),U,DB(1),V
          O'E
              P'T OUT4,DB(0),BIT,STATE,SWV(0),DB(0),U,DB(1),V
              PUNCH FORMAT OUT4,DB(0),BIT,STATE,SWV(0),DB(0),U,
        1     DB(1),V
          E'L
LP1       C'E
         O'E
          W'R SWV(0).E.$      $

        R***** COMPRESS IF NO CONDITIONAL VARIABLES

              P'T OUTU4,B(0),BIT,STATE,B(0),U,B(1),V
              PUNCH FORMAT OUTU4,B(0),BIT,STATE,B(0),U,B(1),V
          O'E
              P'T OUT4,B(0),BIT,STATE,SWV(0),B(0),U,B(1),V
              PUNCH FORMAT OUT4,B(0),BIT,STATE,SWV(0),B(0),U,B(1),V
          E'L
         E'L
LOOP     C'E
         F'N
         V'S VAL2 = $S8,C1,S9,C1,S34,C1*$
         V'S OUT4 = $1H0,C4,C1,3H = ,C5,3H*P*,C5,2(1H*,C6,C1)*$
         V'S OUT6 = $1H0,C6,C1,3H = ,C5,3H*P*,C5,2(1H*,C6,C1)*$
         V'S OUTU4 = $1H0,C4,C1,3H = ,C5,2H*P,2(1H*,C6,C1)*$
         V'S OUTU6 = $1H0,C6,C1,3H = ,C5,2H*P,2(1H*,C6,C1)*$
         E'N
```

```
 EXTERNAL FUNCTION VBL32.(NX)

R***** GENERATE TERMS FOR ADD/SUB

 N'S INTEGER
 P'N STATE,REG(3),SWV(2),FLF(1),B(4),DX(11)
 D'N V(4)
 D'N AB(4),BB(4)
 T'H LOOP, FOR I=0,1, I.G.7
 R'T VAL3,V(0)...V(4)
 U=$ *$

R***** COMPLEMENT FIRST VARIABLE IF TRUTH VALUE 0

 W'R V(0).E.$0$, U=$'*$
 X=$ *$

R***** COMPLEMENT SECOND VARIABLE IF TRUTH VALUE 0

 W'R V(1).E.$0$, X=$'*$
 W=$ $

R***** COMPLEMENT THIRD VARIABLE IF TRUTH VALUE 0

 W'R V(2).E.$0$, W=$'$
 W'R V(0).E.$0$.AND.V(3).E.$1$

R***** TEST FOR J TERM
      BIT=FLF(1)
 O'R V(0).E.$1$.AND.V(3).E.$0$

R***** TEST FOR K TERM
      BIT=FLF(0)
 O'E
      T'O CARRY
 E'L
 W'R B(0).E.$A(I)$

R***** GENERATE TERMS FOR ALL BITS OF A

      MASK=777700777777K
      T'H LP1, FOR J=1,1, J.G.7
      NRA=J.LS.18
      AB(0)=(B(0).A.MASK).V.NRA
      AB(1)=(B(1).A.MASK).V.NRA
      AB(2)=(B(2).A.MASK).V.NRA
      P'T OUTA,AB(0),BIT,STATE,AB(0),U,AB(1),X,AB(2),W
LP1   PUNCH FORMAT OUTA,AB(0),BIT,STATE,AB(0),U,AB(1),X,AB(2),W
```

```
          O'E
              P'T OUTA,B(0),BIT,STATE,B(0),U,B(1),X,B(2),W
              PUNCH FORMAT OUTA,B(0),BIT,STATE,B(0),U,B(1),X,B(2),W
          E'L
CARRY     W'R V(4).E.$1$.AND.B(4).E.$B(I-1)$

         R***** GENERATE CARRY BIT EXPRESSIONS (B(0)...B(7))

              T'H LP2, FOR K=1,1, K.G.8
              MASK=777700777777K
              NRB=K.LS.18
              BB(0)=(B(0).A.MASK).V.NRB
              BB(1)=(B(1).A.MASK).V.NRB
              BB(2)=(B(2).A.MASK).V.NRB
              P'T OUTCY,K-1,STATE,BB(0),U,BB(1),X,BB(2),W
LP2           PUNCH FORMAT OUTCY,K-1,STATE,BB(0),U,BB(1),X,BB(2),W
          O'E
              C'E
          E'L
LOOP      C'E
          W'R STATE.E.$K(11)$.AND.B(0).E.$A(8)$

         R**** GENERATE LAST CARRY BIT EXPRESSION FOR SUBTRACTION-B(8)

              P'T CY8,STATE
              PUNCH FORMAT CY8,STATE
          E'L
          F'N
          V'S VAL3 = $S8,C1,S9,C1,S9,C1,S24,C1,S9,C1*$
          V'S OUTA = $1H0,C4,C1,3H = ,C5,3H*P*,2(C4,C2),C4,C1*$
          V'S OUTCY = $1H0,2HB(,I1,5H)  = ,C5,3H*P*,2(C4,C2),C4,C1*$
          V'S CY8 = $1H0,8HB(8)  = ,C5,2H*P*$
          E'N
```

APPENDIX I

BOOLEAN EXPRESSION PROGRAM OUTPUT

```
OF(0)K = K(17)*P*G
OF(3)K = K(17)*P*G
OF(0)J = K(06)*P*G'
OF(3)J = K(06)*P*G'
OF(0)J = K(06)*P*G
OF(1)K = K(06)*P*G
OF(0)K = K(12)*P
OF(2)K = K(12)*P*R(1)'
OF(0)K = K(12)*P
OF(2)K = K(12)*P*R(1)'
OF(3)J = K(12)*P*R(2)
OF(0)K = K(12)*P
OF(0)K = K(12)*P
OF(3)J = K(12)*P*R(2)
OF(0)K = K(12)*P
OF(1)J = K(12)*P*R(0)
OF(2)K = K(12)*P*R(1)'
OF(0)K = K(12)*P
OF(1)J = K(12)*P*R(0)
OF(2)K = K(12)*P*R(1)'
OF(3)J = K(12)*P*R(2)
OF(0)J = K(00)*P
OF(2)J = K(10)*P
OF(3)J = K(10)*P
OF(0)J = K(01)*P
OF(2)J = K(11)*P
OF(0)J = K(04)*P
OF(1)K = K(04)*P
OF(2)J = K(04)*P
OF(3)J = K(04)*P
OF(0)J = K(02)*P
OF(3)J = K(02)*P
OF(0)J = K(03)*P
OF(0)J = K(05)*P*C(3)
OF(2)J = K(05)*P*C(3)
OF(0)J = K(05)*P*C(2)
OF(2)J = K(05)*P*C(2)
OF(3)K = K(05)*P*C(2)
OF(0)J = K(05)*P*C(1)
OF(0)J = K(05)*P*C(0)
OF(3)K = K(05)*P*C(0)
OF(0)K = K(13)*P
OF(1)J = K(13)*P
OF(3)K = K(13)*P
OF(1)K = K(14)*P
OF(2)J = K(14)*P
OF(3)J = K(14)*P
```

```
OF(1)K = K(15)*P
OF(2)J = K(15)*P
OF(1)K = K(16)*P
OF(3)J = K(16)*P
0  G K = POWER*P*  G
0  G J = START*P*  G '
0  G K = K(05)*P*C(3) *  G
OC(0)K = K(17)*P*    G'*C(0)
OC(1)K = K(17)*P*    G'*C(1)
OC(2)K = K(17)*P*    G'*C(2)
OC(3)K = K(17)*P*    G'*C(3)
OC(4)K = K(17)*P*    G'*C(4)
OC(5)K = K(17)*P*    G'*C(5)
OC(0)J = K(12)*P*C(0)   '*R(3)
OC(1)J = K(12)*P*C(1)   '*R(4)
OC(2)J = K(12)*P*C(2)   '*R(5)
OC(3)J = K(12)*P*C(3)   '*R(6)
OC(4)J = K(12)*P*C(4)   '*R(7)
OC(5)J = K(12)*P*C(5)   '*R(8)
OC(0)K = K(12)*P*C(0)    *R(3)   '
OC(1)K = K(12)*P*C(1)    *R(4)   '
OC(2)K = K(12)*P*C(2)    *R(5)   '
OC(3)K = K(12)*P*C(3)    *R(6)   '
OC(4)K = K(12)*P*C(4)    *R(7)   '
OC(5)K = K(12)*P*C(5)    *R(8)   '
OC(0)J = K(13)*P*C(0)   '*D(0)
OC(1)J = K(13)*P*C(1)   '*D(1)
OC(2)J = K(13)*P*C(2)   '*D(2)
OC(3)J = K(13)*P*C(3)   '*D(3)
OC(4)J = K(13)*P*C(4)   '*D(4)
OC(5)J = K(13)*P*C(5)   '*D(5)
OC(0)K = K(13)*P*C(0)    *D(0)   '
OC(1)K = K(13)*P*C(1)    *D(1)   '
OC(2)K = K(13)*P*C(2)    *D(2)   '
OC(3)K = K(13)*P*C(3)    *D(3)   '
OC(4)K = K(13)*P*C(4)    *D(4)   '
OC(5)K = K(13)*P*C(5)    *D(5)   '
OD(0)K = K(17)*P*    G'*D(0)
OD(1)K = K(17)*P*    G'*D(1)
OD(2)K = K(17)*P*    G'*D(2)
OD(3)K = K(17)*P*    G'*D(3)
OD(4)K = K(17)*P*    G'*D(4)
OD(5)K = K(17)*P*    G'*D(5)
OD(0)J = K(06)*P*D(0)'*D(1)*D(2)*D(3)*D(4)*D(5)
OD(0)K = K(06)*P*D(0)*D(1)*D(2)*D(3)*D(4)*D(5)
OD(1)J = K(06)*P*D(1)'*D(2)*D(3)*D(4)*D(5)
OD(1)K = K(06)*P*D(1)*D(2)*D(3)*D(4)*D(5)
OD(2)J = K(06)*P*D(2)'*D(3)*D(4)*D(5)
OD(2)K = K(06)*P*D(2)*D(3)*D(4)*D(5)
OD(3)J = K(06)*P*D(3)'*D(4)*D(5)
OD(3)K = K(06)*P*D(3)*D(4)*D(5)
OD(4)J = K(06)*P*D(4)'*D(5)
OD(4)K = K(06)*P*D(4)*D(5)
```

```
0D(5)J = K(06)*P*D(5)'
0D(5)K = K(06)*P*D(5)
0D(0)J = K(04)*P*D(0)   '*R(3)
0D(1)J = K(04)*P*D(1)   '*R(4)
0D(2)J = K(04)*P*D(2)   '*R(5)
0D(3)J = K(04)*P*D(3)   '*R(6)
0D(4)J = K(04)*P*D(4)   '*R(7)
0D(5)J = K(04)*P*D(5)   '*R(8)
0D(0)K = K(04)*P*D(0)    *R(3)   '
0D(1)K = K(04)*P*D(1)    *R(4)   '
0D(2)K = K(04)*P*D(2)    *R(5)   '
0D(3)K = K(04)*P*D(3)    *R(6)   '
0D(4)K = K(04)*P*D(4)    *R(7)   '
0D(5)K = K(04)*P*D(5)    *R(8)   '
0D(0)J = K(02)*P*A(0) *D(0)   '*R(3)
0D(1)J = K(02)*P*A(0) *D(1)   '*R(4)
0D(2)J = K(02)*P*A(0) *D(2)   '*R(5)
0D(3)J = K(02)*P*A(0) *D(3)   '*R(6)
0D(4)J = K(02)*P*A(0) *D(4)   '*R(7)
0D(5)J = K(02)*P*A(0) *D(5)   '*R(8)
0D(0)K = K(02)*P*A(0) *D(0)    *R(3)   '
0D(1)K = K(02)*P*A(0) *D(1)    *R(4)   '
0D(2)K = K(02)*P*A(0) *D(2)    *R(5)   '
0D(3)K = K(02)*P*A(0) *D(3)    *R(6)   '
0D(4)K = K(02)*P*A(0) *D(4)    *R(7)   '
0D(5)K = K(02)*P*A(0) *D(5)    *R(8)   '
0R(I)J = K(06)*P*R(I)   '*M(C,I)
0R(I)K = K(06)*P*R(I)    *M(C,I)'
0R(I)J = K(00)*P*R(I)   '*M(C,I)
0R(I)K = K(00)*P*R(I)    *M(C,I)'
0R(I)J = K(01)*P*R(I)   '*M(C,I)
0R(I)K = K(01)*P*R(I)    *M(C,I)'
0M(C,I)J = K(03)*P*M(C,I)'*A(I)
0M(C,I)K = K(03)*P*M(C,I) *A(I)   '
0A(0)J = K(10)*P*A(0)'*R(0)'*B(0)
0A(0)J = K(10)*P*A(0)'*R(0) *B(0)'
0A(0)K = K(10)*P*A(0) *R(0)'*B(0)
0A(0)K = K(10)*P*A(0) *R(0) *B(0)'
0A(1)J = K(10)*P*A(1)'*R(1)'*B(1)
0A(2)J = K(10)*P*A(2)'*R(2)'*B(2)
0A(3)J = K(10)*P*A(3)'*R(3)'*B(3)
0A(4)J = K(10)*P*A(4)'*R(4)'*B(4)
0A(5)J = K(10)*P*A(5)'*R(5)'*B(5)
0A(6)J = K(10)*P*A(6)'*R(6)'*B(6)
0A(7)J = K(10)*P*A(7)'*R(7)'*B(7)
0A(1)J = K(10)*P*A(1)'*R(1) *B(1)'
0A(2)J = K(10)*P*A(2)'*R(2) *B(2)'
0A(3)J = K(10)*P*A(3)'*R(3) *B(3)'
0A(4)J = K(10)*P*A(4)'*R(4) *B(4)'
0A(5)J = K(10)*P*A(5)'*R(5) *B(5)'
0A(6)J = K(10)*P*A(6)'*R(6) *B(6)'
0A(7)J = K(10)*P*A(7)'*R(7) *B(7)'
```

```
0A(0)J = K(11)*P*A(0)'*R(0)'*B(0)'
0A(0)J = K(11)*P*A(0)'*R(0) *B(0)
0A(0)K = K(11)*P*A(0) *R(0)'*B(0)'
0A(0)K = K(11)*P*A(0) *R(0) *B(0)
0A(1)J = K(11)*P*A(1)'*R(1)'*B(1)'
0A(2)J = K(11)*P*A(2)'*R(2)'*B(2)'
0A(3)J = K(11)*P*A(3)'*R(3)'*B(3)'
0A(4)J = K(11)*P*A(4)'*R(4)'*B(4)'
0A(5)J = K(11)*P*A(5)'*R(5)'*B(5)'
0A(6)J = K(11)*P*A(6)'*R(6)'*B(6)'
0A(7)J = K(11)*P*A(7)'*R(7)'*B(7)'
0B(0)  = K(11)*P*A(1)'*R(1)'*B(1)
0B(1)  = K(11)*P*A(2)'*R(2)'*B(2)
0B(2)  = K(11)*P*A(3)'*R(3)'*B(3)
0B(3)  = K(11)*P*A(4)'*R(4)'*B(4)
0B(4)  = K(11)*P*A(5)'*R(5)'*B(5)
0B(5)  = K(11)*P*A(6)'*R(6)'*B(6)
0B(6)  = K(11)*P*A(7)'*R(7)'*B(7)
0B(7)  = K(11)*P*A(8)'*R(8)'*B(8)
0A(1)J = K(11)*P*A(1)'*R(1) *B(1)
0A(2)J = K(11)*P*A(2)'*R(2) *B(2)
0A(3)J = K(11)*P*A(3)'*R(3) *B(3)
0A(4)J = K(11)*P*A(4)'*R(4) *B(4)
0A(5)J = K(11)*P*A(5)'*R(5) *B(5)
0A(6)J = K(11)*P*A(6)'*R(6) *B(6)
0A(7)J = K(11)*P*A(7)'*R(7) *B(7)
0A(1)K = K(11)*P*A(1) *R(1)'*B(1)'
0A(2)K = K(11)*P*A(2) *R(2)'*B(2)'
0A(3)K = K(11)*P*A(3) *R(3)'*B(3)'
0A(4)K = K(11)*P*A(4) *R(4)'*B(4)'
0A(5)K = K(11)*P*A(5) *R(5)'*B(5)'
0A(6)K = K(11)*P*A(6) *R(6)'*B(6)'
0A(7)K = K(11)*P*A(7) *R(7)'*B(7)'
0B(0)  = K(11)*P*A(1) *R(1)'*B(1)'
0B(1)  = K(11)*P*A(2) *R(2)'*B(2)'
0B(2)  = K(11)*P*A(3) *R(3)'*B(3)'
0B(3)  = K(11)*P*A(4) *R(4)'*B(4)'
0B(4)  = K(11)*P*A(5) *R(5)'*B(5)'
0B(5)  = K(11)*P*A(6) *R(6)'*B(6)'
0B(6)  = K(11)*P*A(7) *R(7)'*B(7)'
0B(7)  = K(11)*P*A(8) *R(8)'*B(8)'
0B(0)  = K(11)*P*A(1) *R(1)'*B(1)
0B(1)  = K(11)*P*A(2) *R(2)'*B(2)
0B(2)  = K(11)*P*A(3) *R(3)'*B(3)
0B(3)  = K(11)*P*A(4) *R(4)'*B(4)
0B(4)  = K(11)*P*A(5) *R(5)'*B(5)
0B(5)  = K(11)*P*A(6) *R(6)'*B(6)
0B(6)  = K(11)*P*A(7) *R(7)'*B(7)
0B(7)  = K(11)*P*A(8) *R(8)'*B(8)
```

```
0A(1)K = K(11)*P*A(1) *R(1) *B(1)
0A(2)K = K(11)*P*A(2) *R(2) *B(2)
0A(3)K = K(11)*P*A(3) *R(3) *B(3)
0A(4)K = K(11)*P*A(4) *R(4) *B(4)
0A(5)K = K(11)*P*A(5) *R(5) *B(5)
0A(6)K = K(11)*P*A(6) *R(6) *B(6)
0A(7)K = K(11)*P*A(7) *R(7) *B(7)
0B(0)  = K(11)*P*A(1) *R(1) *B(1)
0B(1)  = K(11)*P*A(2) *R(2) *B(2)
0B(2)  = K(11)*P*A(3) *R(3) *B(3)
0B(3)  = K(11)*P*A(4) *R(4) *B(4)
0B(4)  = K(11)*P*A(5) *R(5) *B(5)
0B(5)  = K(11)*P*A(6) *R(6) *B(6)
0B(6)  = K(11)*P*A(7) *R(7) *B(7)
0B(7)  = K(11)*P*A(8) *R(8) *B(8)
0A(8)J = K(11)*P*A(8)'*R(8)'*B(8)'
0A(8)J = K(11)*P*A(8)'*R(8) *B(8)
0A(8)K = K(11)*P*A(8) *R(8)'*B(8)'
0A(8)K = K(11)*P*A(8) *R(8) *B(8)
0B(8)  = K(11)*P
0A(1)J = K(14)*P*A(1)  '*A(0)
0A(2)J = K(14)*P*A(2)  '*A(1)
0A(3)J = K(14)*P*A(3)  '*A(2)
0A(4)J = K(14)*P*A(4)  '*A(3)
0A(5)J = K(14)*P*A(5)  '*A(4)
0A(6)J = K(14)*P*A(6)  '*A(5)
0A(7)J = K(14)*P*A(7)  '*A(6)
0A(1)K = K(14)*P*A(1)   *A(0)  '
0A(2)K = K(14)*P*A(2)   *A(1)  '
0A(3)K = K(14)*P*A(3)   *A(2)  '
0A(4)K = K(14)*P*A(4)   *A(3)  '
0A(5)K = K(14)*P*A(5)   *A(4)  '
0A(6)K = K(14)*P*A(6)   *A(5)  '
0A(7)K = K(14)*P*A(7)   *A(6)  '
0A(8)J = K(14)*P*A(8)  '*A(7)
0A(8)K = K(14)*P*A(8)   *A(7)  '
0A(0)J = K(15)*P*A(0)  '*A(1)
0A(0)K = K(15)*P*A(0)   *A(1)  '
0A(1)J = K(15)*P*A(1)  '*A(2)
0A(2)J = K(15)*P*A(2)  '*A(3)
0A(3)J = K(15)*P*A(3)  '*A(4)
0A(4)J = K(15)*P*A(4)  '*A(5)
0A(5)J = K(15)*P*A(5)  '*A(6)
0A(6)J = K(15)*P*A(6)  '*A(7)
0A(7)J = K(15)*P*A(7)  '*A(8)
0A(1)K = K(15)*P*A(1)   *A(2)  '
0A(2)K = K(15)*P*A(2)   *A(3)  '
0A(3)K = K(15)*P*A(3)   *A(4)  '
0A(4)K = K(15)*P*A(4)   *A(5)  '
0A(5)K = K(15)*P*A(5)   *A(6)  '
0A(6)K = K(15)*P*A(6)   *A(7)  '
0A(7)K = K(15)*P*A(7)   *A(8)  '
0A(8)J = K(15)*P*A(8)  '*A(0)
0A(8)K = K(15)*P*A(8)   *A(0)  '
```

```
OA(0)K = K(16)*P*A(0)
OA(1)K = K(16)*P*A(1)
OA(2)K = K(16)*P*A(2)
OA(3)K = K(16)*P*A(3)
OA(4)K = K(16)*P*A(4)
OA(5)K = K(16)*P*A(5)
OA(6)K = K(16)*P*A(6)
OA(7)K = K(16)*P*A(7)
OA(8)K = K(16)*P*A(8)
```

APPENDIX J

BOOLEAN EQUATION PROGRAM

```
          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N T(500)
          V'S EQUAL = $=  $
          V'S PLUS = $ + $

         R***** LOAD F EXPRESSIONS

          T'H LOOP1, FOR A=0,1, A.G.500
          L'T IN,JK
          W'R (JK.A.77K10).NE.$F00000$, T'O EXIT1
          R'T IN,JK,ST,CP,V
          BIT(A)=JK
          STATE(A)=ST
          CLOCK(A)=CP
LOOP1     VAR(A)=V
EXIT1     EXECUTE FEQN.(A)

         R***** LOAD G EXPRESSIONS

          T'H LOOP2, FOR B=1,1, B.G.100
          L'T IN,JK
          W'R (JK.A.77K6).NE.$00G000$, T'O EXIT2
          LOAD.(B,T)
LOOP2     C'E
EXIT2     EXECUTE GEQN.(B)

         R***** LOAD C EXPRESSIONS

          T'H LOOP3, FOR C=1,1, C.G.100
          L'T IN,JK
          W'R (JK.A.77K10).NE.$C00000$, T'O EXIT3
          LOAD.(C,T)
LOOP3     C'E
EXIT3     EXECUTE CEQN.(C)

         R***** LOAD D EXPRESSIONS

          T'H LOOP4, FOR D=1,1, D.G.100
          L'T IN,JK
          W'R (JK.A.77K10).NE.$D00000$, T'O EXIT4
          LOAD.(D,T)
LOOP4     C'E
EXIT4     EXECUTE DEQN.(D)
```

```
          R***** LOAD R EXPRESSIONS

           T'H LOOP5, FOR E=1,1, E.G.100
           L'T IN,JK
           W'R (JK.A.77K10).NE.$R00000$, T'O EXIT5
           LOAD.(E,T)
LOOP5      C'E
EXIT5      EXECUTE REQN.(E)

          R***** LOAD M EXPRESSIONS

           T'H LOOP6, FOR F=1,1, F.G.100
           L'T IN,JK
           W'R (JK.A.77K10).NE.$M00000$, T'O EXIT6
           LOAD.(F,T)
LOOP6      C'E
EXIT6      EXECUTE MEQN.(F,T)

          R***** LOAD A EXPRESSIONS

           T'H LOOP7, FOR G=1,1, G.G.500
           L'T IN,JK
           W'R (JK.A.77K10).NE.$A00000$.AND.(JK.A.77K10).NE.$B00000$,
          1T'O EXIT7
           LOAD.(G,T)
LOOP7      C'E
EXIT7      EXECUTE AEQN.(G)
           V'S OUT = $1H0,C5,S3,3(C3,5C6)/1H0,S8,3(C3,5C6)*$
           V'S CHECK = $1H0,S5,20HCHECK INPUT SEQUENCE*$
           V'S IN = $S1,C5,S3,C5,C3,C5*$
           E'M
```

```
          EXTERNAL FUNCTION FEQN.(X)

         R***** GENERATE F REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N F(8*100),N(8)
          V'S N=0,3,3,3,3,3,3,3,3

         R***** MINIMIZE EXPRESSIONS

          T'H LOOP, FOR I=0,1, I.G.X-2
          T'H LOOP, FOR J=I+1,1, J.G.X-1
          W'R BIT(I).E.BIT(J).AND.STATE(I).E.STATE(J)
              W'R VAR(I).E.VAR(J)
                  BIT(J)=$      $
                  STATE(J)=$      $
                  CLOCK(J)=$    $
                  VAR(J)=$      $
              O'R (VAR(I).EV.VAR(J)).E.74K8.OR.(VAR(I).EV.VAR(J)).E.
         1    74K2
                  BIT(J)=$      $
                  STATE(J)=$      $
                  CLOCK(J)=$    $
                  VAR(J)=$      $
                  CLOCK(I)=$*P  $
                  VAR(I)=$      $
              E'L
          O'E
              C'E
LOOP      E'L

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          F(1,1)=$F(0)J$
          F(2,1)=$F(0)K$
          F(3,1)=$F(1)J$
          F(4,1)=$F(1)K$
          F(5,1)=$F(2)J$
          F(6,1)=$F(2)K$
          F(7,1)=$F(3)J$
          F(8,1)=$F(3)K$
          T'H LP1, FOR K=1,1, K.G.8
LP1       F(K,2)=EQUAL
          T'H LP2, FOR L=0,1, L.G.X-1
          T'H LP3, FOR M=1,1, M.G.8
LP3       W'R BIT(L).E.F(M,1), T'O PACK
          T'O LP2
```

```
OB(0)   = K(10)*P*A(1)'*R(1) *B(1)
OB(1)   = K(10)*P*A(2)'*R(2) *B(2)
OB(2)   = K(10)*P*A(3)'*R(3) *B(3)
OB(3)   = K(10)*P*A(4)'*R(4) *B(4)
OB(4)   = K(10)*P*A(5)'*R(5) *B(5)
OB(5)   = K(10)*P*A(6)'*R(6) *B(6)
OB(6)   = K(10)*P*A(7)'*R(7) *B(7)
OB(7)   = K(10)*P*A(8)'*R(8) *B(8)
OA(1)K  = K(10)*P*A(1) *R(1)'*B(1)
OA(2)K  = K(10)*P*A(2) *R(2)'*B(2)
OA(3)K  = K(10)*P*A(3) *R(3)'*B(3)
OA(4)K  = K(10)*P*A(4) *R(4)'*B(4)
OA(5)K  = K(10)*P*A(5) *R(5)'*B(5)
OA(6)K  = K(10)*P*A(6) *R(6)'*B(6)
OA(7)K  = K(10)*P*A(7) *R(7)'*B(7)
OB(0)   = K(10)*P*A(1) *R(1)'*B(1)
OB(1)   = K(10)*P*A(2) *R(2)'*B(2)
OB(2)   = K(10)*P*A(3) *R(3)'*B(3)
OB(3)   = K(10)*P*A(4) *R(4)'*B(4)
OB(4)   = K(10)*P*A(5) *R(5)'*B(5)
OB(5)   = K(10)*P*A(6) *R(6)'*B(6)
OB(6)   = K(10)*P*A(7) *R(7)'*B(7)
OB(7)   = K(10)*P*A(8) *R(8)'*B(8)
OA(1)K  = K(10)*P*A(1) *R(1) *B(1)'
OA(2)K  = K(10)*P*A(2) *R(2) *B(2)'
OA(3)K  = K(10)*P*A(3) *R(3) *B(3)'
OA(4)K  = K(10)*P*A(4) *R(4) *B(4)'
OA(5)K  = K(10)*P*A(5) *R(5) *B(5)'
OA(6)K  = K(10)*P*A(6) *R(6) *B(6)'
OA(7)K  = K(10)*P*A(7) *R(7) *B(7)'
OB(0)   = K(10)*P*A(1) *R(1) *B(1)'
OB(1)   = K(10)*P*A(2) *R(2) *B(2)'
OB(2)   = K(10)*P*A(3) *R(3) *B(3)'
OB(3)   = K(10)*P*A(4) *R(4) *B(4)'
OB(4)   = K(10)*P*A(5) *R(5) *B(5)'
OB(5)   = K(10)*P*A(6) *R(6) *B(6)'
OB(6)   = K(10)*P*A(7) *R(7) *B(7)'
OB(7)   = K(10)*P*A(8) *R(8) *B(8)'
OB(0)   = K(10)*P*A(1) *R(1) *B(1)
OB(1)   = K(10)*P*A(2) *R(2) *B(2)
OB(2)   = K(10)*P*A(3) *R(3) *B(3)
OB(3)   = K(10)*P*A(4) *R(4) *B(4)
OB(4)   = K(10)*P*A(5) *R(5) *B(5)
OB(5)   = K(10)*P*A(6) *R(6) *B(6)
OB(6)   = K(10)*P*A(7) *R(7) *B(7)
OB(7)   = K(10)*P*A(8) *R(8) *B(8)
OA(8)J  = K(10)*P*A(8)'*R(8)'*B(8)
OA(8)J  = K(10)*P*A(8)'*R(8) *B(8)'
OA(8)K  = K(10)*P*A(8) *R(8)'*B(8)
OA(8)K  = K(10)*P*A(8) *R(8) *B(8)'
```

```
          R***** PACK EXPRESSIONS INTO EQUATIONS

PACK           F(M,N(M))=STATE(L)
               F(M,(N(M)+1))=CLOCK(L)
               F(M,(N(M)+2))=VAR(L)
               W'R N(M).G.3, F(M,(N(M)-1))=PLUS
               N(M)=N(M)+4
LP2             C'E
                T'H LP4, FOR P=1,2, P.G.7
                    F(P,N(P))=$POWER$
                    F(P,(N(P)-1))=PLUS
                    F((P+1),N(P+1))=$      $
LP4                 F((P+1),(N(P+1)-1))=$    $
                P'T $1H1*$

          R***** PRINT EQUATIONS

                T'H LP5, FOR B=1,1, B.G.8
                P'T $1H *$
LP5             P'T EQNS,F(B,1)...F(B,N(B))
                F'N
                V'S EQNS = $1H0,C5,S2,14(C3,C5)/1H0,S7,14(C3,C5)*$
                E'N
```

```
          EXTERNAL FUNCTION LOAD.(I,T)

         R***** LOAD EXPRESSION MATRIX

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N VX(41),VY(6)
          TT=$ $
          L'T IN1,JK
          W'R (JK.A.77K10).E.$M00000$
              R'T IN2,JK,TT,VX(0)...VX(41)
          O'E
              R'T IN1,JK,VX(0)...VX(41)
          E'L
          CMPCK.(VX,VY)
          BIT(I)=JK
          T(I)=TT
          T'H LOOP, FOR J=1,1, J.G.7
LOOP      Y(J,I)=VY(J-1)
          F'N
          V'S IN1 = $S1,C5,S3,42C1*$
          V'S IN2 = $S1,C6,C1,S3,42C1*$
          E'N



          EXTERNAL FUNCTION CMPCK.(X,Y)

         R***** COMPRESS AND PACK INPUT EXPRESSIONS

          N'S INTEGER
          D'N A(41),B(5)
          N=0
          T'H LOOP, FOR I=0,1, I.G.41
          W'R X(I).NE.$ $
              A(N)=X(I)
              N=N+1
          O'E
              C'E
LOOP      E'L
          T'H LP1, FOR J=N,1, J.G.41
LP1       A(J)=$ $
          T'H LP2, FOR K=0,1, K.G.6
          Y(K)=$000000$
          T'H LP2, FOR M=0,1, M.G.5
          B(M)=(A((K*6)+M).A.77K10).RS.(6*M)
LP2       Y(K)=Y(K).V.B(M)
          F'N
          E'N
```

```
          EXTERNAL FUNCTION GEQN.(X)

         R***** GENERATE G REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N G(2*100),NG(2),GC(2)
          V'S NG(1)=3,3
          V'S GC(1)=$  G J$,$  G K$

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          G(1,1)=$G(0)J$
          G(1,2)=EQUAL
          G(2,1)=$G(0)K$
          G(2,2)=EQUAL
          T'H LOOP, FOR I=1,1, I.G.X-1
          T'H LP1, FOR J=1,1, J.G.2
LP1       W'R BIT(I).E.GC(J), T'O PACK
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACK       W'R Y(1,I).E.$POWER*$.OR.Y(1,I).E.$START*$
               G(J,NG(J))=(Y(1,I).A.7777777777K2).V.$00000 $
           T'H LP2, FOR K=1,1, K.G.4
LP2            G(J,NG(J)+K)=$       $
           O'E
               T'H LP3, FOR L=0,1, L.G.4
LP3            G(J,NG(J)+L)=Y(L+1,I)
           E'L
           W'R NG(J).G.3, G(J,NG(J)-1)=PLUS
           NG(J)=NG(J)+6
LOOP       C'E
           P'T $1H1*$

         R***** PRINT EQUATIONS

          P'T OUT,G(1,1)...G(1,NG(1)-2)
          P'T OUT,G(2,1)...G(2,NG(2)-2)
          F'N
          E'N
```

```
          EXTERNAL FUNCTION MEQN.(X,T)

         R***** GENERATE MEMORY WORD EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N M(2*100),NM(2)
          V'S NM(1)=4,4

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          M(1,1)=$M(C,I)$
          M(2,1)=$M(C,I)$
          M(1,2)=$J$
          M(2,2)=$K$
          M(1,3)=EQUAL
          M(2,3)=EQUAL
          T'H LOOPM, FOR I=1,1, I.G.X-1
          T'H LPM1, FOR J=1,1, J.G.2
LPM1      W'R BIT(I).E.M(J,1).AND.T(I).E.M(J,2), T'O PACKM
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACKM     T'H LPM2, FOR K=0,1, K.G.4
LPM2      M(J,NM(J)+K)=Y(K+1,I)
          W'R NM(J).G.4, M(J,NM(J)-1)=PLUS
          NM(J)=NM(J)+6
LOOPM     C'E
          P'T $1H2*$

         R***** PRINT EQUATIONS

          P'T OUTM,M(1,1)...M(1,NM(1)-2)
          P'T OUTM,M(2,1)...M(2,NM(2)-2)
          PRINT COMMENT $0      I = 0,...,8$
          F'N
          V'S OUTM = $1H-,C6,C1,S3,3(C3,5C6)/1H0,S10,3(C3,5C6)*$
          E'N
```

```
          EXTERNAL FUNCTION DEQN.(B)

         R***** GENERATE D REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N D(12*100),ND(12)
          V'S ND(1)=3,3,3,3,3,3,3,3,3,3,3,3

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          T'H LP1, FOR I=0,1, I.G.5
          D(2*I+2,1)=$D(0)K$.V.(I.LS.18)
          D(2*I+1,1)=$D(0)J$.V.(I.LS.18)
          D(2*I+1,2)=EQUAL
LP1       D(2*I+2,2)=EQUAL
          T'H LOOP, FOR J=1,1, J.G.B-1
          T'H LP2, FOR K=1,1, K.G.12
LP2       W'R BIT(J).E.D(K,1), T'O PACK
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACK      T'H LP3, FOR L=0,1, L.G.6
LP3       D(K,ND(K)+L)=Y(L+1,J)
          W'R ND(K).G.3, D(K,ND(K)-1)=PLUS
          ND(K)=ND(K)+8
LOOP      C'E
          P'T $1H1*$

         R***** PRINT EQUATIONS

          T'H LP4, FOR M=1,1, M.G.12
LP4       P'T OUTD,D(M,1)...D(M,ND(M)-2)
          F'N
          V'S OUTD = $1H-,C5,S3,2(C3,7C6)/1H0,S8,2(C3,7C6)*$
          E'N
```

```
          EXTERNAL FUNCTION CEQN.(B)

         R***** GENERATE C REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N C(12*100),NC(12)
          V'S NC(1)=3,3,3,3,3,3,3,3,3,3,3,3

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          T'H LP1, FOR I=0,1, I.G.5
          C(2*I+1,1)=$C(0)J$.V.(I.LS.18)
          C(2*I+2,1)=$C(0)K$.V.(I.LS.18)
          C(2*I+1,2)=EQUAL
LP1       C(2*I+2,2)=EQUAL
          T'H LOOP, FOR J=1,1, J.G.B-1
          T'H LP2, FOR K=1,1, K.G.12
LP2       W'R BIT(J).E.C(K,1), T'O PACK
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACK      T'H LP3, FOR L=0,1, L.G.4
LP3       C(K,NC(K)+L)=Y(L+1,J)
          W'R NC(K).G.3, C(K,NC(K)-1)=PLUS
          NC(K)=NC(K)+6
LOOP      C'E
          P'T $1H1*$

         R***** PRINT EQUATIONS

          T'H LP4, FOR M=1,1, M.G.12
          P'T $1H *$
LP4       P'T OUT,C(M,1)...C(M,NC(M)-2)
          F'N
          E'N
```

```
          EXTERNAL FUNCTION REQN.(X)

         R***** GENERATE R REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N R(2*100),NR(2)
          V'S NR(1)=3,3

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          R(1,1)=$R(I)J$
          R(2,1)=$R(I)K$
          R(1,2)=EQUAL
          R(2,2)=EQUAL
          T'H LOOP, FOR I=1,1, I.G.X-1
          T'H LPR1, FOR J=1,1, J.G.2
LPR1      W'R BIT(I).E.R(J,1), T'O PACK
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACK      T'H LPR2, FOR K=0,1, K.G.4
LPR2      R(J,NR(J)+K)=Y(K+1,I)
          W'R NR(J).G.3, R(J,NR(J)-1)=PLUS
          NR(J)=NR(J)+6
LOOP      C'E
          P'T $1H1*$

         R***** PRINT EQUATIONS

          P'T OUT,R(1,1)...R(1,NR(1)-2)
          P'T OUT,R(2,1)...R(2,NR(2)-2)
          PRINT COMMENT $0      I = 0,...,8$
          F'N
          E'N
```

```
          EXTERNAL FUNCTION AEQN.(X)

         R***** GENERATE A REGISTER EQUATIONS

          N'S INTEGER
          P'N EQUAL,PLUS,OUT(6),CHECK(5),BIT(500),Y(7*500),STATE(100),
         1CLOCK(100),VAR(100)
          D'N A(18*100),NA(18),B(9*100),NB(9)

         R***** LOAD EQUATIONS MATRIX ROW NAMES

          T'H LP1, FOR I=0,1, I.G.8
          A(2*I+1,1)=$A(0)J$.V.(I.LS.18)
          A(2*I+2,1)=$A(0)K$.V.(I.LS.18)
          B(I+1,1)=$B(0)$.V.(I.LS.18)
          A(2*I+1,2)=EQUAL
          A(2*I+2,2)=EQUAL
          B(I+1,2)=EQUAL
          NA(2*I+1)=3
          NA(2*I+2)=3
LP1       NB(I+1)=3
          T'H LOOP, FOR J=1,1, J.G.X-1
          T'H LP2, FOR K=1,1, K.G.18
LP2       W'R BIT(J).E.A(K,1), T'O PACKA
          T'H LP3, FOR L=1,1, L.G.9
LP3       W'R BIT(J).E.B(L,1), T'O PACKB
          P'T CHECK
          ERROR RETURN

         R***** PACK EXPRESSIONS INTO EQUATIONS

PACKA     T'H LP4, FOR M=0,1, M.G.4
LP4       A(K,NA(K)+M)=Y(M+1,J)
          W'R NA(K).G.3, A(K,NA(K)-1)=PLUS
          NA(K)=NA(K)+6
          T'O LOOP
PACKB     T'H LP5, FOR N=0,1, N.G.4
LP5       B(L,NB(L)+N)=Y(N+1,J)
          W'R NB(L).G.3, B(L,NB(L)-1)=PLUS
          NB(L)=NB(L)+6
LOOP      C'E
          P'T $1H1*$

         R***** PRINT EQUATIONS

          T'H LP6, FOR P=1,1, P.G.18
LP6       P'T OUTA,A(P,1)...A(P,NA(P)-2)
          T'H LP7, FOR Q=1,1, Q.G.9
LP7       P'T OUTA,B(Q,1)...B(Q,NB(Q)-2)
          F'N
          V'S OUTA = $1H-,C5,S3,3(C3,5C6)/1H0,S8,3(C3,5C6)/1H0,
         1S8,3(C3,5C6)*$
          E'N
```

APPENDIX K

BOOLEAN EQUATION PROGRAM OUTPUT

```
F(0)J  =  K(06)*P      + K(00)*P      + K(01)*P      + K(04)*P      + K(02)*P      + K(03)*P      + K(05)*P*C(3)
        + K(05)*P*C(2) + K(05)*P*C(1) + K(05)*P*C(0) + POWER

F(0)K  =  K(17)*P*G    + K(12)*P      + K(13)*P

F(1)J  =  K(12)*P*R(0) + K(13)*P      + POWER

F(1)K  =  K(06)*P*G    + K(04)*P      + K(14)*P      + K(15)*P      + K(16)*P

F(2)J  =  K(10)*P      + K(11)*P      + K(04)*P      + K(05)*P*C(3) + K(05)*P*C(2) + K(14)*P      + K(15)*P
        + POWER

F(2)K  =  K(12)*P*R(1)'

F(3)J  =  K(06)*P*G'   + K(12)*P*R(2) + K(10)*P      + K(04)*P      + K(02)*P      + K(14)*P      + K(16)*P
        + POWER

F(3)K  =  K(17)*P*G    + K(05)*P*C(2) + K(05)*P*C(0) + K(13)*P
```

```
G(0)J   =  START
G(0)K   =  POWER                    + K(05)*P*C(3)*G
```

```
C(0)J   =  K(12)*P*C(0)'*R(3)          + K(13)*P*C(0)'*D(0)

C(0)K   =  K(17)*P*G'*C(0)             + K(12)*P*C(0)*R(3)'          + K(13)*P*C(0)*D(0)'


C(1)J   =  K(12)*P*C(1)'*R(4)          + K(13)*P*C(1)'*D(1)

C(1)K   =  K(17)*P*G'*C(1)             + K(12)*P*C(1)*R(4)'          + K(13)*P*C(1)*D(1)'


C(2)J   =  K(12)*P*C(2)'*R(5)          + K(13)*P*C(2)'*D(2)

C(2)K   =  K(17)*P*G'*C(2)             + K(12)*P*C(2)*R(5)'          + K(13)*P*C(2)*D(2)'


C(3)J   =  K(12)*P*C(3)'*R(6)          + K(13)*P*C(3)'*D(3)

C(3)K   =  K(17)*P*G'*C(3)             + K(12)*P*C(3)*R(6)'          + K(13)*P*C(3)*D(3)'


C(4)J   =  K(12)*P*C(4)'*R(7)          + K(13)*P*C(4)'*D(4)

C(4)K   =  K(17)*P*G'*C(4)             + K(12)*P*C(4)*R(7)'          + K(13)*P*C(4)*D(4)'


C(5)J   =  K(12)*P*C(5)'*R(8)          + K(13)*P*C(5)'*D(5)
```

```
D(0)J  =  K(06)*P*D(0)'*D(1)*D(2)*D(3)*D(4)*D(5)      + K(04)*P*D(0)'*R(3)
        + K(02)*P*A(0)*D(0)'*R(3)

D(0)K  =  K(17)*P*G'*D(0)                            + K(06)*P*D(0)*D(1)*D(2)*D(3)*D(4)*D(5)
        + K(04)*P*D(0)*R(3)'                         + K(02)*P*A(0)*D(0)*R(3)'

D(1)J  =  K(06)*P*D(1)'*D(2)*D(3)*D(4)*D(5)          + K(04)*P*D(1)'*R(4)
        + K(02)*P*A(0)*D(1)'*R(4)

D(1)K  =  K(17)*P*G'*D(1)                            + K(06)*P*D(1)*D(2)*D(3)*D(4)*D(5)
        + K(04)*P*D(1)*R(4)'                         + K(02)*P*A(0)*D(1)*R(4)'

D(2)J  =  K(06)*P*D(2)'*D(3)*D(4)*D(5)               + K(04)*P*D(2)'*R(5)
        + K(02)*P*A(0)*D(2)'*R(5)

D(2)K  =  K(17)*P*G'*D(2)                            + K(06)*P*D(2)*D(3)*D(4)*D(5)
        + K(04)*P*D(2)*R(5)'                         + K(02)*P*A(0)*D(2)*R(5)'

D(3)J  =  K(06)*P*D(3)'*D(4)*D(5)                    + K(04)*P*D(3)'*R(6)
        + K(02)*P*A(0)*D(3)'*R(6)

D(3)K  =  K(17)*P*G'*D(3)                            + K(06)*P*D(3)*D(4)*D(5)
        + K(04)*P*D(3)*R(6)'                         + K(02)*P*A(0)*D(3)*R(6)'

D(4)J  =  K(06)*P*D(4)'*D(5)                         + K(04)*P*D(4)'*R(7)
        + K(02)*P*A(0)*D(4)'*R(7)

D(4)K  =  K(17)*P*G'*D(4)                            + K(06)*P*D(4)*D(5)
        + K(04)*P*D(4)*R(7)'                         + K(02)*P*A(0)*D(4)*R(7)'

D(5)J  =  K(06)*P*D(5)'                              + K(04)*P*D(5)'*R(8)
        + K(02)*P*A(0)*D(5)'*R(8)

D(5)K  =  K(17)*P*G'*D(5)                            + K(06)*P*D(5)
        + K(04)*P*D(5)*R(8)'                         + K(02)*P*A(0)*D(5)*R(8)'
```

$$R(I)J = K(06)*P*R(I)'*M(C,I) + K(00)*P*R(I)'*M(C,I) + K(01)*P*R(I)'*M(C,I)$$

$$R(I)K = K(06)*P*R(I)*M(C,I)' + K(00)*P*R(I)*M(C,I)' + K(01)*P*R(I)*M(C,I)'$$

$$I = 0,\ldots,8$$

$$M(C,I)J = K(03)*P*M(C,I)'*A(I)$$

$$M(C,I)K = K(03)*P*M(C,I)*A(I)'$$

$$I = 0,\ldots,8$$

```
          + K(11)*P*A(0)'*R(0)*B(0)        + K(15)*P*A(0)'*A(1)

A(0)K  =  K(10)*P*A(0)*R(0)'*B(0)          + K(10)*P*A(0)*R(0)*B(0)'          + K(11)*P*A(0)*R(0)'*B(0)'
          + K(11)*P*A(0)*R(0)*B(0)         + K(15)*P*A(0)*A(1)'               + K(16)*P*A(0)

A(1)J  =  K(10)*P*A(1)'*R(1)'*B(1)         + K(10)*P*A(1)'*R(1)*B(1)'         + K(11)*P*A(1)'*R(1)'*B(1)'
          + K(11)*P*A(1)'*R(1)*B(1)        + K(14)*P*A(1)'*A(0)               + K(15)*P*A(1)'*A(2)

A(1)K  =  K(10)*P*A(1)*R(1)'*B(1)          + K(10)*P*A(1)*R(1)*B(1)'          + K(11)*P*A(1)*R(1)'*B(1)'
          + K(11)*P*A(1)*R(1)*B(1)         + K(14)*P*A(1)*A(0)'               + K(15)*P*A(1)*A(2)'
          + K(16)*P*A(1)

A(2)J  =  K(10)*P*A(2)'*R(2)'*B(2)         + K(10)*P*A(2)'*R(2)*B(2)'         + K(11)*P*A(2)'*R(2)'*B(2)'
          + K(11)*P*A(2)'*R(2)*B(2)        + K(14)*P*A(2)'*A(1)               + K(15)*P*A(2)'*A(3)

A(2)K  =  K(10)*P*A(2)*R(2)'*B(2)          + K(10)*P*A(2)*R(2)*B(2)'          + K(11)*P*A(2)*R(2)'*B(2)'
          + K(11)*P*A(2)*R(2)*B(2)         + K(14)*P*A(2)*A(1)'               + K(15)*P*A(2)*A(3)'
          + K(16)*P*A(2)

A(3)J  =  K(10)*P*A(3)'*R(3)'*B(3)         + K(10)*P*A(3)'*R(3)*B(3)'         + K(11)*P*A(3)'*R(3)'*B(3)'
          + K(11)*P*A(3)'*R(3)*B(3)        + K(14)*P*A(3)'*A(2)               + K(15)*P*A(3)'*A(4)

A(3)K  =  K(10)*P*A(3)*R(3)'*B(3)          + K(10)*P*A(3)*R(3)*B(3)'          + K(11)*P*A(3)*R(3)'*B(3)'
          + K(11)*P*A(3)*R(3)*B(3)         + K(14)*P*A(3)*A(2)'               + K(15)*P*A(3)*A(4)'
          + K(16)*P*A(3)

A(4)J  =  K(10)*P*A(4)'*R(4)'*B(4)         + K(10)*P*A(4)'*R(4)*B(4)'         + K(11)*P*A(4)'*R(4)'*B(4)'
          + K(11)*P*A(4)'*R(4)*B(4)        + K(14)*P*A(4)'*A(3)               + K(15)*P*A(4)'*A(5)
```

```
A(4)K  =  K(10)*P*A(4)*R(4)'*B(4)        + K(10)*P*A(4)*R(4)*B(4)'        + K(11)*P*A(4)*R(4)'*B(4)'
        + K(11)*P*A(4)*R(4)*B(4)         + K(14)*P*A(4)*A(3)'             + K(15)*P*A(4)*A(5)'
        + K(16)*P*A(4)

A(5)J  =  K(10)*P*A(5)'*R(5)'*B(5)       + K(10)*P*A(5)'*R(5)*B(5)'       + K(11)*P*A(5)'*R(5)'*B(5)'
        + K(11)*P*A(5)'*R(5)*B(5)        + K(14)*P*A(5)'*A(4)             + K(15)*P*A(5)'*A(6)

A(5)K  =  K(10)*P*A(5)*R(5)'*B(5)        + K(10)*P*A(5)*R(5)*B(5)'        + K(11)*P*A(5)*R(5)'*B(5)'
        + K(11)*P*A(5)*R(5)*B(5)         + K(14)*P*A(5)*A(4)'             + K(15)*P*A(5)*A(6)'
        + K(16)*P*A(5)

A(6)J  =  K(10)*P*A(6)'*R(6)'*B(6)       + K(10)*P*A(6)'*R(6)*B(6)'       + K(11)*P*A(6)'*R(6)'*B(6)'
        + K(11)*P*A(6)'*R(6)*B(6)        + K(14)*P*A(6)'*A(5)             + K(15)*P*A(6)'*A(7)

A(6)K  =  K(10)*P*A(6)*R(6)'*B(6)        + K(10)*P*A(6)*R(6)*B(6)'        + K(11)*P*A(6)*R(6)'*B(6)'
        + K(11)*P*A(6)*R(6)*B(6)         + K(14)*P*A(6)*A(5)'             + K(15)*P*A(6)*A(7)'
        + K(16)*P*A(6)

A(7)J  =  K(10)*P*A(7)'*R(7)'*B(7)       + K(10)*P*A(7)'*R(7)*B(7)'       + K(11)*P*A(7)'*R(7)'*B(7)'
        + K(11)*P*A(7)'*R(7)*B(7)        + K(14)*P*A(7)'*A(6)             + K(15)*P*A(7)'*A(8)

A(7)K  =  K(10)*P*A(7)*R(7)'*B(7)        + K(10)*P*A(7)*R(7)*B(7)'        + K(11)*P*A(7)*R(7)'*B(7)'
        + K(11)*P*A(7)*R(7)*B(7)         + K(14)*P*A(7)*A(6)'             + K(15)*P*A(7)*A(8)'
        + K(16)*P*A(7)

A(8)J  =  K(10)*P*A(8)'*R(8)'*B(8)       + K(10)*P*A(8)'*R(8)*B(8)'       + K(11)*P*A(8)'*R(8)'*B(8)'
        + K(11)*P*A(8)'*R(8)*B(8)        + K(14)*P*A(8)'*A(7)             + K(15)*P*A(8)'*A(0)

A(8)K  =  K(10)*P*A(8)*R(8)'*B(8)        + K(10)*P*A(8)*R(8)*B(8)'        + K(11)*P*A(8)*R(8)'*B(8)'
```

```
          + K(11)*P*A(8)*R(8)*B(8)      + K(14)*P*A(8)*A(7)'          + K(15)*P*A(8)*A(0)'
          + K(16)*P*A(8)

B(0)   =  K(10)*P*A(1)'*R(1)*B(1)       + K(10)*P*A(1)*R(1)'*B(1)     + K(10)*P*A(1)*R(1)*B(1)'
          + K(10)*P*A(1)*R(1)*B(1)      + K(11)*P*A(1)'*R(1)'*B(1)    + K(11)*P*A(1)*R(1)'*B(1)'
          + K(11)*P*A(1)*R(1)'*B(1)     + K(11)*P*A(1)*R(1)*B(1)

B(1)   =  K(10)*P*A(2)'*R(2)*B(2)       + K(10)*P*A(2)*R(2)'*B(2)     + K(10)*P*A(2)*R(2)*B(2)'
          + K(10)*P*A(2)*R(2)*B(2)      + K(11)*P*A(2)'*R(2)'*B(2)    + K(11)*P*A(2)*R(2)'*B(2)'
          + K(11)*P*A(2)*R(2)'*B(2)     + K(11)*P*A(2)*R(2)*B(2)

B(2)   =  K(10)*P*A(3)'*R(3)*B(3)       + K(10)*P*A(3)*R(3)'*B(3)     + K(10)*P*A(3)*R(3)*B(3)'
          + K(10)*P*A(3)*R(3)*B(3)      + K(11)*P*A(3)'*R(3)'*B(3)    + K(11)*P*A(3)*R(3)'*B(3)'
          + K(11)*P*A(3)*R(3)'*B(3)     + K(11)*P*A(3)*R(3)*B(3)

B(3)   =  K(10)*P*A(4)'*R(4)*B(4)       + K(10)*P*A(4)*R(4)'*B(4)     + K(10)*P*A(4)*R(4)*B(4)'
          + K(10)*P*A(4)*R(4)*B(4)      + K(11)*P*A(4)'*R(4)'*B(4)    + K(11)*P*A(4)*R(4)'*B(4)'
          + K(11)*P*A(4)*R(4)'*B(4)     + K(11)*P*A(4)*R(4)*B(4)

B(4)   =  K(10)*P*A(5)'*R(5)*B(5)       + K(10)*P*A(5)*R(5)'*B(5)     + K(10)*P*A(5)*R(5)*B(5)'
          + K(10)*P*A(5)*R(5)*B(5)      + K(11)*P*A(5)'*R(5)'*B(5)    + K(11)*P*A(5)*R(5)'*B(5)'
          + K(11)*P*A(5)*R(5)'*B(5)     + K(11)*P*A(5)*R(5)*B(5)

B(5)   =  K(10)*P*A(6)'*R(6)*B(6)       + K(10)*P*A(6)*R(6)'*B(6)     + K(10)*P*A(6)*R(6)*B(6)'
          + K(10)*P*A(6)*R(6)*B(6)      + K(11)*P*A(6)'*R(6)'*B(6)    + K(11)*P*A(6)*R(6)'*B(6)'
          + K(11)*P*A(6)*R(6)'*B(6)     + K(11)*P*A(6)*R(6)*B(6)

B(6)   =  K(10)*P*A(7)'*R(7)*B(7)       + K(10)*P*A(7)*R(7)'*B(7)     + K(10)*P*A(7)*R(7)*B(7)'
          + K(10)*P*A(7)*R(7)*B(7)      + K(11)*P*A(7)'*R(7)'*B(7)    + K(11)*P*A(7)*R(7)'*B(7)'
          + K(11)*P*A(7)*R(7)'*B(7)     + K(11)*P*A(7)*R(7)*B(7)

B(7)   =  K(10)*P*A(8)'*R(8)*B(8)       + K(10)*P*A(8)*R(8)'*B(8)     + K(10)*P*A(8)*R(8)*B(8)'
          + K(10)*P*A(8)*R(8)*B(8)      + K(11)*P*A(8)'*R(8)'*B(8)    + K(11)*P*A(8)*R(8)'*B(8)'
          + K(11)*P*A(8)*R(8)'*B(8)     + K(11)*P*A(8)*R(8)*B(8)

B(8)   =  K(11)*P
```

BIBLIOGRAPHY

Books

Chu, Yaohan. Digital Computer Design Fundamentals. New York, San Francisco, Toronto, London: McGraw-Hill Book Company (1962).

Other Sources

Arden, B., Galler B., and Graham, R. The Michigan Algorithmic Decoder. University of Michigan (April, 1965).

Chu, Yaohan. "An ALGOL-like Computer Design Language," Communications of the ACM, VIII, Number 10 (October, 1965).